Synchronization Architecture in Parallel Programming Models

PhD Thesis

Arturo González Escribano

Supervisors: Valentín Cardeñoso Payo (Univ. Valladolid)
Arie J.C. van Gemund (TU Delft)

July, 2003

Outline

Introduction

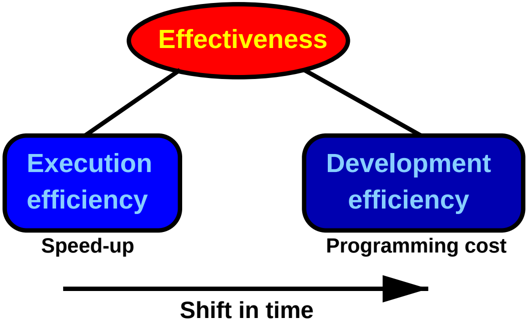
Conceptual approach and models review

Theoretical and algorithmic approach

Experimental approach

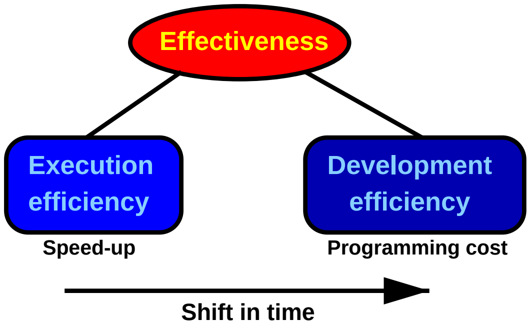
Conclusion

Motivation



[Siegel2000] [Smith2001] [Danelutto2003]

Motivation

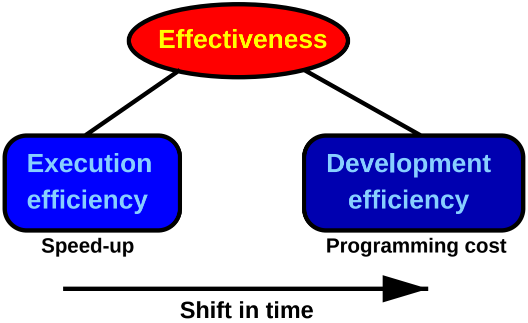


[Siegel2000] [Smith2001] [Danelutto2003]

- No well-established parallel computing model or reference architecture

  [SkillicornTalia98]

Motivation



[Siegel2000] [Smith2001] [Danelutto2003]

- No well-established parallel computing model or reference architecture [SkillicornTalia98]
- Lack of a *Parallel Programming Model (PPM)* which achieves both:
  - Software development capabilities
  - Portability and efficient implementations

Parallel systems modeling challenges

Existing approaches

Parallel systems modeling challenges

Existing approaches

- High abstract level
  - + Elegant semantic models
  - – Complex specifications
  - – Too far from lower level details for easy implementation

Parallel systems modeling challenges

Existing approaches

- High abstract level
  - + Elegant semantic models
  - – Complex specifications
  - – Too far from lower level details for easy implementation

- Focused on the low level details
  - + Allow to exploit all parallelism power
  - – Difficult to program, analyze and debug

Parallel systems modeling challenges

Existing approaches

- High abstract level
  - \+ Elegant semantic models
  - – Complex specifications
  - – Too far from lower level details for easy implementation
- Focused on the low level details
  - \+ Allow to exploit all parallelism power
  - – Difficult to program, analyze and debug
- Restricted models
  - – Reduce the expressive power
  - \+ Simple & analyzable structures

Parallel systems modeling challenges

Existing approaches

- High abstract level
  - + Elegant semantic models
  - – Complex specifications
  - – Too far from lower level details for easy implementation

- Focused on the low level details
  - + Allow to exploit all parallelism power
  - – Difficult to program, analyze and debug

- Restricted models
  - – Reduce the expressive power
  - + Simple & analyzable structures

The expressive power and analyzability of a model appear to be highly related to communication/synchronization

SA

- We coin the term Synchronization Architecture (SA) to summarize the formal description of communication & synchronization logic structures

SA

- We coin the term Synchronization Architecture (SA) to summarize the formal description of communication & synchronization logic structures
- Why architecture?
  - Description of synchronization/communication mechanisms
  - Description of the composition rules

SA

- We coin the term Synchronization Architecture (SA) to summarize the formal description of communication & synchronization logic structures

- Why architecture?
  - Description of synchronization/communication mechanisms
  - Description of the composition rules

- Why synchronization?
  - Generalization of both communication & synchronization

SA

- We coin the term Synchronization Architecture (SA) to summarize the formal description of communication & synchronization logic structures

- Why architecture?
  - Description of synchronization/communication mechanisms
  - Description of the composition rules

- Why synchronization?
  - Generalization of both communication & synchronization
  - Synchronization and computation are orthogonal

    [GelernterCarriero92]

SA

- We coin the term Synchronization Architecture (SA) to summarize the formal description of communication & synchronization logic structures

- Why architecture?
  - Description of synchronization/communication mechanisms
  - Description of the composition rules

- Why synchronization?
  - Generalization of both communication & synchronization
  - Synchronization and computation are orthogonal

    [GelernterCarriero92]
  - Synchronization distinguishes parallel from sequential solutions

Problem statement

- What is the relationship between SA and properties of PPMs?

  We propose a new classification system for PPMs

Problem statement

- What is the relationship between SA and properties of PPMs?

  We propose a new classification system for PPMs

- What are the advantages and drawbacks of restricted SAs?

  We show that one SA class, called SP, groups the most interesting models

Problem statement

- What is the relationship between SA and properties of PPMs?

  We propose a new classification system for PPMs

- What are the advantages and drawbacks of restricted SAs?

  We show that one SA class, called SP, groups the most interesting models

- How is expressive power affected by the restriction?

  We present systematic transformation methods to map non-SP applications into SP form

  We investigate the potential performance impact of these transformations

Problem statement

- What is the relationship between SA and properties of PPMs?

  We propose a new classification system for PPMs

- What are the advantages and drawbacks of restricted SAs?

  We show that one SA class, called SP, groups the most interesting models

- How is expressive power affected by the restriction?

  We present systematic transformation methods to map non-SP applications into SP form

  We investigate the potential performance impact of these transformations

We will show that SP PPMs bring a good trade-off between expressive power and analyzability, being a good choice for general-purpose parallel computing

Approach

Three-step approach

Introduction

Approach

Three-step approach

- Conceptual
  - SA classification
  - Review of models at different abstraction levels
  - Relate SA to PPM characteristics
  - Detect which applications naturally map to each class

Approach

Introduction

Three-step approach

- Conceptual
  - SA classification
  - Review of models at different abstraction levels
  - Relate SA to PPM characteristics
  - Detect which applications naturally map to each class
- Theoretical
  - SP graph characterization
  - NSP to SP transformation (algorithmic) techniques
  - Potential performance loss study

Approach

Three-step approach

- Conceptual
  - SA classification
  - Review of models at different abstraction levels
  - Relate SA to PPM characteristics
  - Detect which applications naturally map to each class
- Theoretical
  - SP graph characterization
  - NSP to SP transformation (algorithmic) techniques
  - Potential performance loss study
- Experimental
  - Graph modeling of applications
  - Experiments with synthetic graphs
  - Experiments with real application graphs

SA classification criteria

Conceptual

SA classification criteria

Conceptual

- Two main types of synchronization [AndrewsSchneider82]

SA classification criteria

Conceptual

- Two main types of synchronization [AndrewsSchneider82]
  - Condition synchronization (CS)
    - ∗ Communications or event synchronization
    - ∗ Implies an execution order

SA classification criteria

- Two main types of synchronization [AndrewsSchneider82]
  - Condition synchronization (CS)
    - Communications or event synchronization
    - Implies an execution order
  - Mutual Exclusion (ME)
    - No concurrent execution, but order is not predefined
    - Final order selection is delayed for lower-level optimization

Conceptual

SA classification criteria

- Two main types of synchronization [AndrewsSchneider82]
  - Condition synchronization (CS)
    - Communications or event synchronization
    - Implies an execution order
  - Mutual Exclusion (ME)
    - No concurrent execution, but order is not predefined
    - Final order selection is delayed for lower-level optimization
  - Orthogonality: A PPM may support one or both of them

SA classification criteria

- Two main types of synchronization [AndrewsSchneider82]
  - Condition synchronization (CS)
    - Communications or event synchronization
    - Implies an execution order
  - Mutual Exclusion (ME)
    - No concurrent execution, but order is not predefined
    - Final order selection is delayed for lower-level optimization
  - Orthogonality: A PPM may support one or both of them
- Classes:

SA classification criteria

Conceptual

- Two main types of synchronization [AndrewsSchneider82]
  - Condition synchronization (CS)
    - ∗ Communications or event synchronization
    - ∗ Implies an execution order
  - Mutual Exclusion (ME)
    - ∗ No concurrent execution, but order is not predefined
    - ∗ Final order selection is delayed for lower-level optimization
  - Orthogonality: A PPM may support one or both of them
- Classes:
  - ME axis: ME vs. NME

SA classification criteria

Conceptual

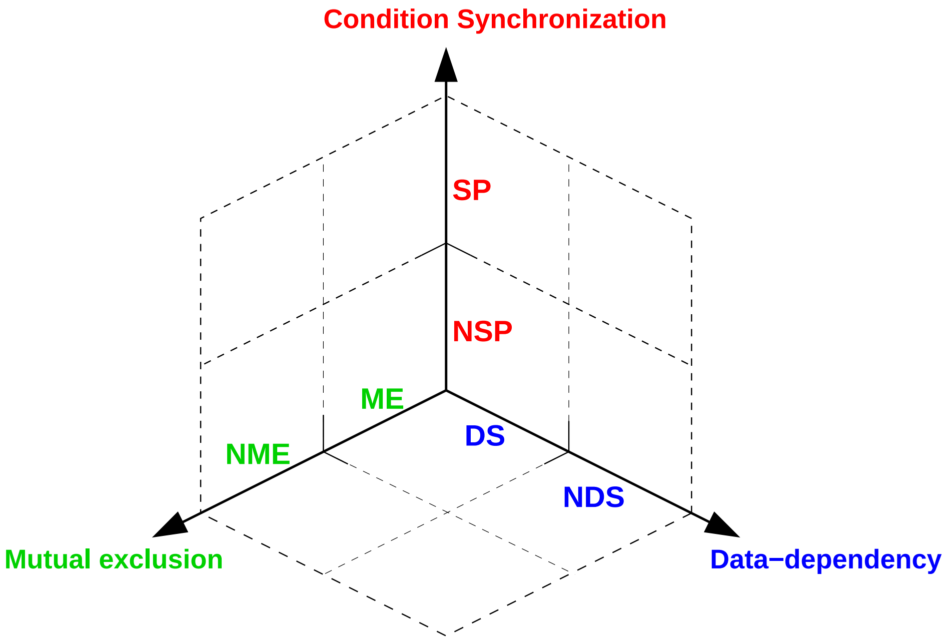
- Two main types of synchronization [AndrewsSchneider82]
  - Condition synchronization (CS)
    - Communications or event synchronization
    - Implies an execution order
  - Mutual Exclusion (ME)
    - No concurrent execution, but order is not predefined
    - Final order selection is delayed for lower-level optimization
  - Orthogonality: A PPM may support one or both of them
- Classes:
  - ME axis: ME vs. NME
  - CS axis: SP (nested-parallelism, cobegin-coend) vs. NSP

SA classification criteria

Conceptual

- Two main types of synchronization [AndrewsSchneider82]
  - Condition synchronization (CS)
    - Communications or event synchronization
    - Implies an execution order
  - Mutual Exclusion (ME)
    - No concurrent execution, but order is not predefined
    - Final order selection is delayed for lower-level optimization
  - Orthogonality: A PPM may support one or both of them
- Classes:
  - ME axis: ME vs. NME
  - CS axis: SP (nested-parallelism, cobegin-coend) vs. NSP
- Data-dependent or dynamic structures (DS)
  - Dynamic conditions and data-dependent synchronizations
  - Impact on analyzability properties [SkillicornTalia98]
  - DS axis: DS vs. NDS

SA classification



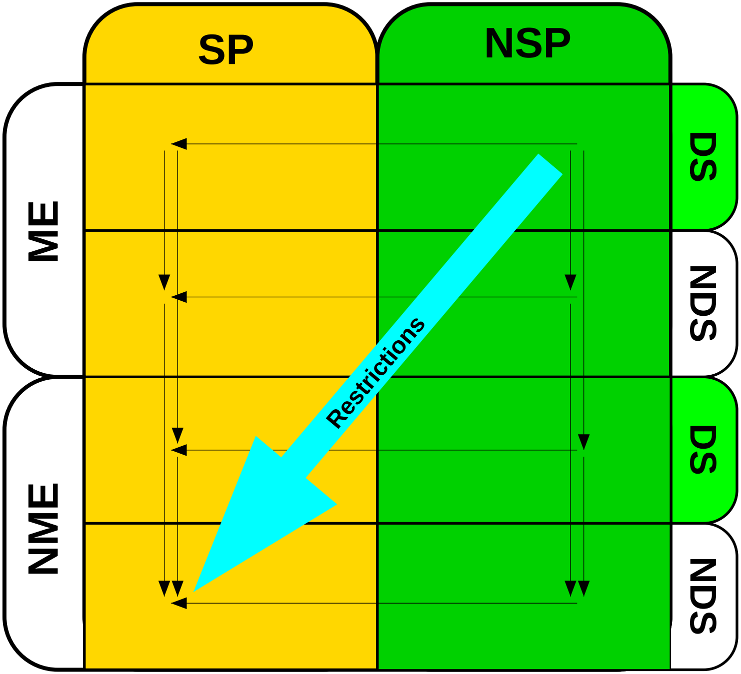
Conceptual

SA classification

Conceptual

		SP	NSP
ME	DS	(SP,ME,DS)	(NSP,ME,DS)
ME	NDS	(SP,ME,NDS)	(NSP,ME,NDS)
NME	DS	(SP,NME,DS)	(NSP,NME,DS)
NME	NDS	(SP,NME,NDS)	(NSP,NME,NDS)

SA classification



Conceptual

Model requirements

Conceptual

Model requirements

- Requirements
  [SkillicornTalia98]

Model requirements

- Requirements
  [SkillicornTalia98]

1. Easy to program
2. Software development technology
3. Easy to understand
4. Architecture independent
5. Cost measures
6. Guaranteed performance

Model requirements

- Requirements [SkillicornTalia98]

1. Easy to program
2. Software development technology
3. Easy to understand
4. Architecture independent
5. Cost measures
6. Guaranteed performance

Model requirements

Conceptual

- Requirements
  [SkillicornTalia98]

1. Easy to program
2. Software development technology
3. Easy to understand
4. Architecture independent
5. Cost measures
6. Guaranteed performance

Model requirements

Conceptual

- Requirements [SkillicornTalia98]

  1 Easy to program
  2 Software development technology
  3 Easy to understand
  4 Architecture independent
  5 Cost measures
  6 Guaranteed performance

- Qualitative and difficult to measure

Model requirements

Conceptual

- Requirements
  [SkillicornTalia98]

1	Easy to program
2	Software development technology
3	Easy to understand
4	Architecture independent
5	Cost measures
6	Guaranteed performance

- Qualitative and difficult to measure

- Quantitative study of performance is possible [JuurlinkWijshof98]

Model requirements

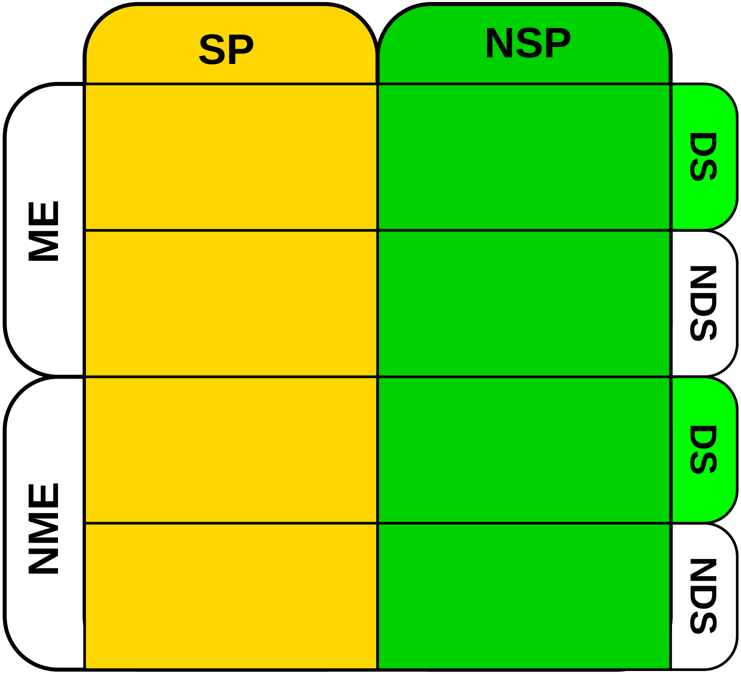
Conceptual

- Requirements
  [SkillicornTalia98]

1. Easy to program
2. Software development technology
3. Easy to understand
4. Architecture independent
5. Cost measures
6. Guaranteed performance

- Qualitative and difficult to measure
- Quantitative study of performance is possible [JuurlinkWijshof98]
- In this work:
  - Review of models to determine adequacy and relate it to SA
  - For the most relevant SA classes, comparative performance study

Classification discussion



Conceptual

Classification discussion

Conceptual

		SP	NSP
ME	DS	Nested–BSP	LogP Tuple–spaces Message–passing
ME	NDS	BSP* OpenMP*	
NME	DS	Cilk	
NME	NDS	PRAM	Skeletons Data–parallelism

Classification discussion

		SP	NSP
ME	DS	Nested−BSP	LogP Tuple−spaces Message−passing
ME	NDS	BSP* OpenMP*	
NME	DS	Cilk	
NME	NDS	PRAM	Skeletons Data−parallelism

Classification discussion

Conceptual

		SP	NSP
ME	DS	Nested−BSP	LogP Tuple−spaces Message−passing
ME	NDS	BSP* OpenMP*	
NME	DS	Cilk	
NME	NDS	PRAM	Skeletons Data−parallelism

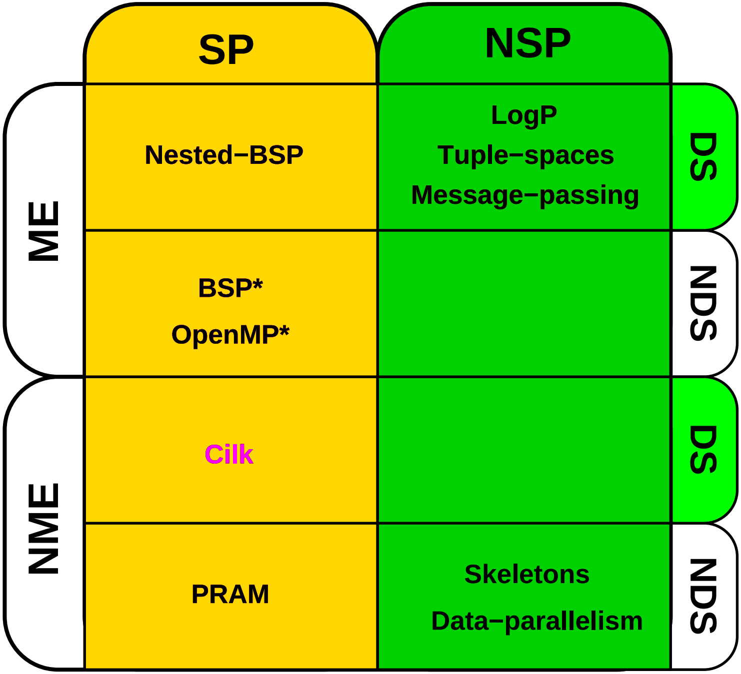
Classification discussion

Conceptual

		SP	NSP
ME	DS	Nested−BSP	LogP Tuple−spaces Message−passing
ME	NDS	BSP* OpenMP*	
NME	DS	Cilk	
NME	NDS	PRAM	Skeletons Data−parallelism

Classification discussion

Conceptual



Classification discussion

Conceptual

	SP	NSP	
ME	Nested−BSP	LogP Tuple−spaces Message−passing	DS
ME	BSP* OpenMP*		NDS
NME	Cilk		DS
NME	PRAM	Skeletons Data−parallelism	NDS

Classification discussion

Conceptual

	SP	NSP	
ME	Nested−BSP	LogP Tuple−spaces Message−passing	DS
ME	BSP* OpenMP*		NDS
NME	Cilk		DS
NME	PRAM	Skeletons Data−parallelism	NDS

Conceptual approach summary

Conceptual

- The SA classification is an adequate categorization of PPMs

Conceptual approach summary

Conceptual

- The SA classification is an adequate categorization of PPMs
- The most adequate models are in the SP class

Conceptual approach summary

- The SA classification is an adequate categorization of PPMs
- The most adequate models are in the SP class
- Study of applications [Dissertation 2.6]

Conceptual approach summary

Conceptual

- The SA classification is an adequate categorization of PPMs
- The most adequate models are in the SP class
- Study of applications [Dissertation 2.6]
  - Many typical applications naturally map to SP
  - Some important classes do not!

Conceptual approach summary

Conceptual

- The SA classification is an adequate categorization of PPMs
- The most adequate models are in the SP class
- Study of applications [Dissertation 2.6]
  - Many typical applications naturally map to SP
  - Some important classes do not!
- Map NSP applications to SP form:

Conceptual

Conceptual approach summary

- The SA classification is an adequate categorization of PPMs
- The most adequate models are in the SP class
- Study of applications [Dissertation 2.6]
  - Many typical applications naturally map to SP
  - Some important classes do not!
- Map NSP applications to SP form:
  Transformations

Conceptual approach summary

Conceptual

- The SA classification is an adequate categorization of PPMs
- The most adequate models are in the SP class
- Study of applications [Dissertation 2.6]
  - Many typical applications naturally map to SP
  - Some important classes do not!
- Map NSP applications to SP form:
  Transformations $\Rightarrow$ Performance loss

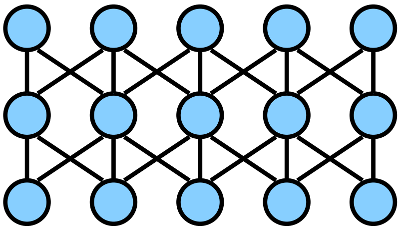
Conceptual approach summary (Example)

Conceptual

Cellular-Automata computation

Conceptual approach summary (Example)

Cellular-Automata computation

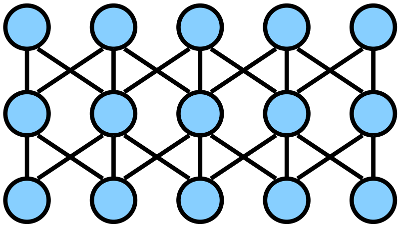


NSP version

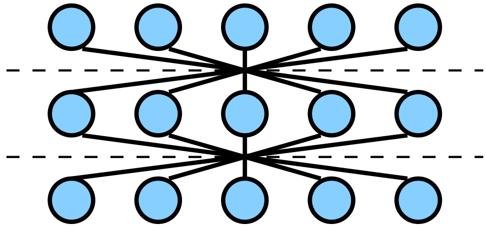
Conceptual approach summary (Example)

Conceptual

Cellular-Automata computation



NSP version

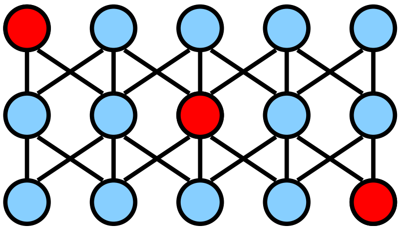


SP version

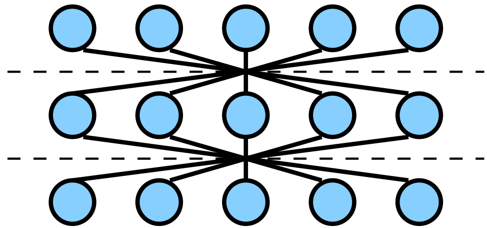
Conceptual approach summary (Example)

Conceptual

Cellular-Automata computation



NSP version

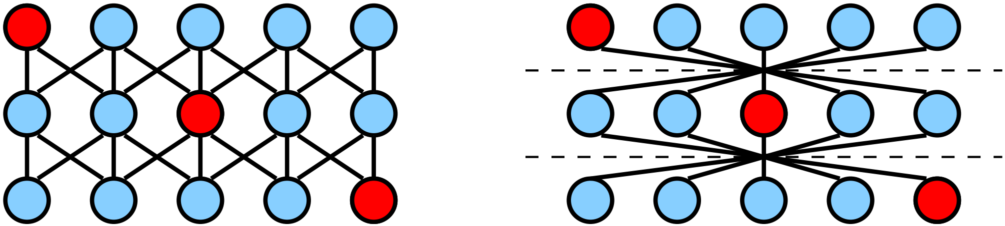


SP version

Conceptual approach summary (Example)

Conceptual

Cellular-Automata computation



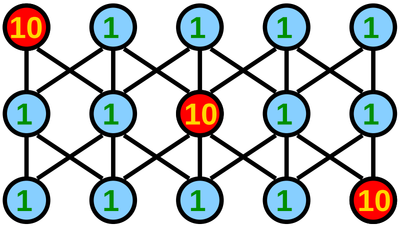
NSP version

SP version

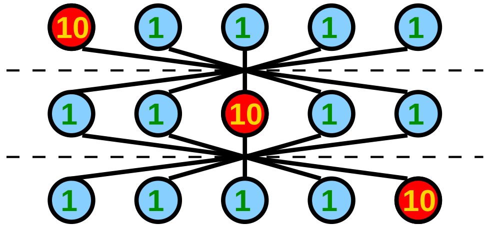
Conceptual approach summary (Example)

Conceptual

Cellular-Automata computation



NSP version

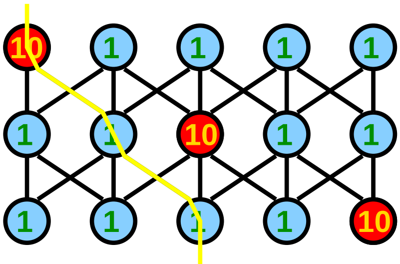


SP version

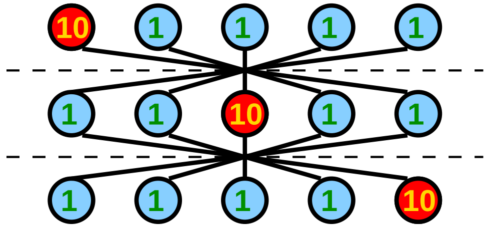
Conceptual approach summary (Example)

Conceptual

Cellular-Automata computation



NSP version

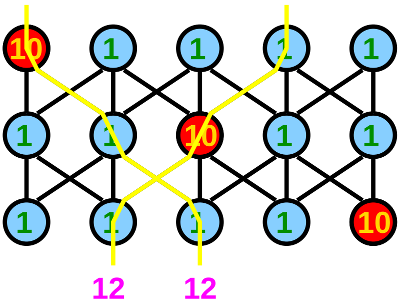


SP version

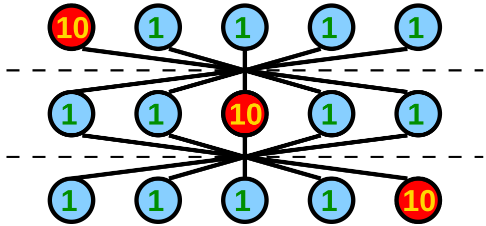
Conceptual approach summary (Example)

Conceptual

Cellular-Automata computation



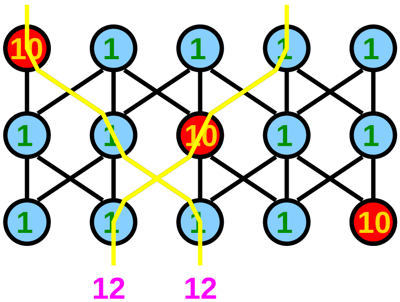
NSP version



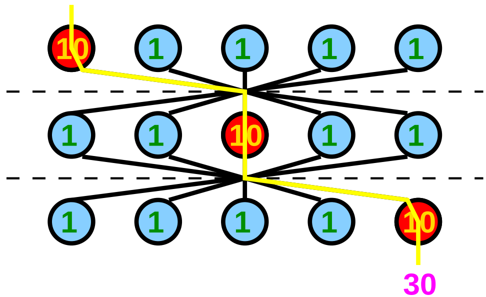
SP version

Conceptual approach summary (Example)

Cellular-Automata computation



NSP version

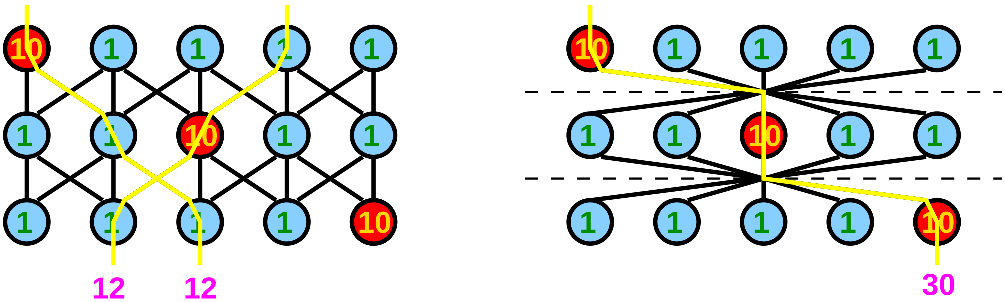


SP version

Conceptual approach summary (Example)

Conceptual

Cellular-Automata computation



NSP version **SP version**

It is necessary to study the transformations: NSP $\longrightarrow$ SP
and their potential performance impact

Theoretical approach

Theoretical

Graphs

Theoretical approach

Graphs

- Formal language: Graph theory

Theoretical approach

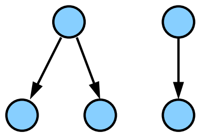
Graphs

- Formal language: Graph theory
- STDAGs (Standard two-terminal direct acyclic graphs)

Theoretical approach

Graphs

- Formal language: Graph theory
- STDAGs (Standard two-terminal direct acyclic graphs)

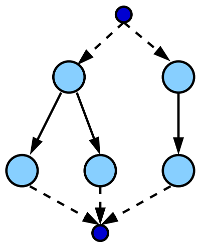


Theoretical approach

Theoretical

Graphs

- Formal language: Graph theory
- STDAGs (Standard two-terminal direct acyclic graphs)

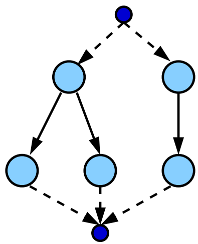


Theoretical approach

Theoretical

Graphs

- Formal language: Graph theory
- STDAGs (Standard two-terminal direct acyclic graphs)



- Modelization of a parallel computation structures with a graph:
  - AoN (Activity on Nodes)
  - Edges: Condition synchronization (execution order)

SP graph

- Compositional recursive definition: [Valdes.et al92]

Theoretical

SP graph

- Compositional recursive definition: [Valdes.et al92]

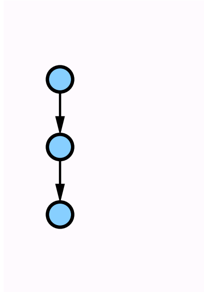
SP graph

- Compositional recursive definition: [Valdes.et al92]

Theoretical

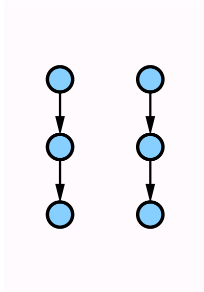
SP graph

- Compositional recursive definition: [Valdes.et al92]



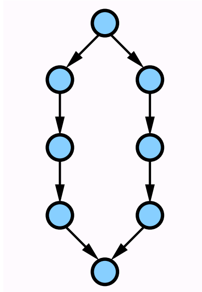
SP graph

- Compositional recursive definition: [Valdes.et al92]



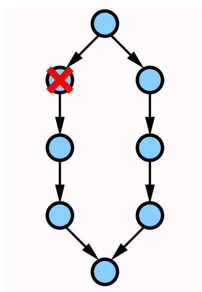
SP graph

- Compositional recursive definition: [Valdes.et al92]



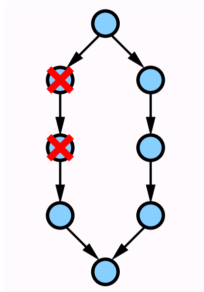
SP graph

- Compositional recursive definition: [Valdes.et al92]



SP graph

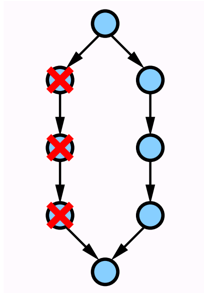
- Compositional recursive definition: [Valdes.et al92]



SP graph

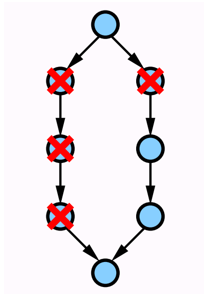
Theoretical

- Compositional recursive definition: [Valdes.et al92]



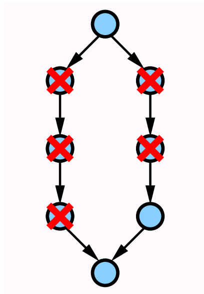
SP graph

- Compositional recursive definition: [Valdes.et al92]



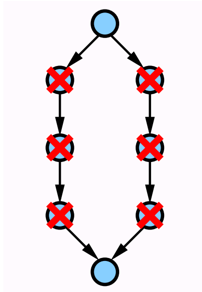
SP graph

- Compositional recursive definition: [Valdes.et al92]



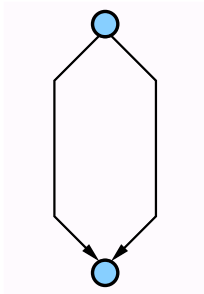
SP graph

- Compositional recursive definition: [Valdes.et al92]



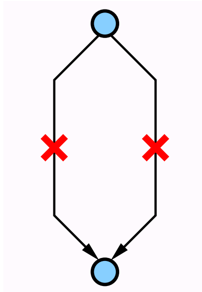
SP graph

- Compositional recursive definition: [Valdes.et al92]



SP graph

- Compositional recursive definition: [Valdes.et al92]

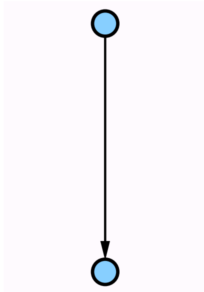


Theoretical

SP graph

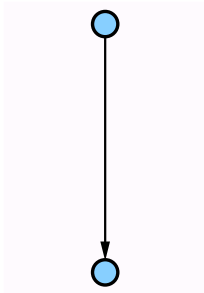
Theoretical

- Compositional recursive definition: [Valdes.et al92]



SP graph

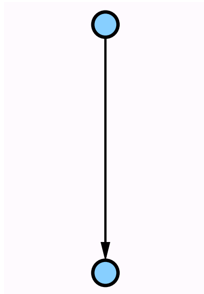
- Compositional recursive definition: [Valdes.et al92]



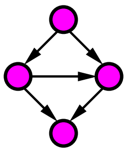
- Forbidden subgraph characterization [Duffin65]

SP graph

- Compositional recursive definition: [Valdes.et al92]



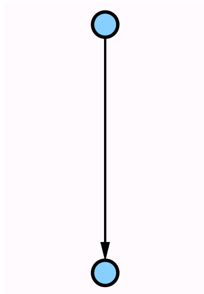
- Forbidden subgraph characterization [Duffin65]



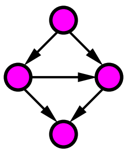
SP graph

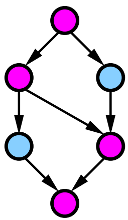
Theoretical

- Compositional recursive definition: [Valdes.et al92]



- Forbidden subgraph characterization [Duffin65]

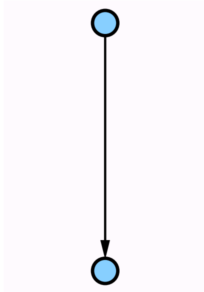




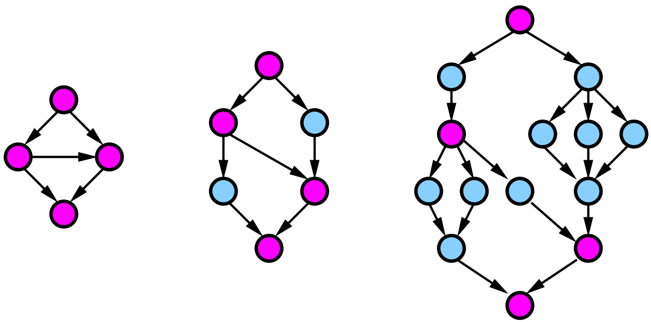
SP graph

Theoretical

- Compositional recursive definition: [Valdes.et al92]



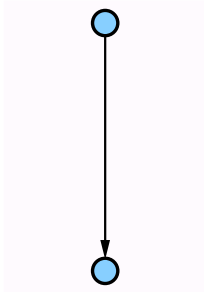
- Forbidden subgraph characterization [Duffin65]



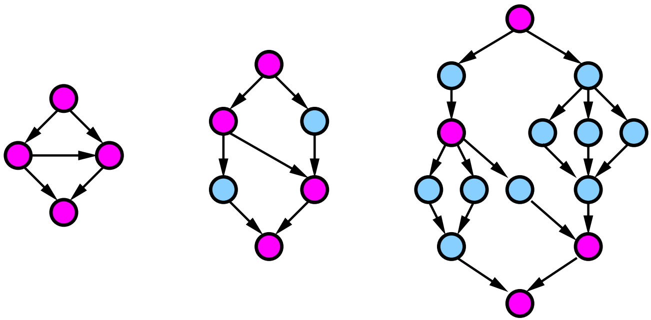
SP graph

Theoretical

- Compositional recursive definition: [Valdes.et al92]



- Forbidden subgraph characterization [Duffin65]
- NSP: Combinations of forbidden subgraphs [Dissertation 3.3.3]



Transformation strategies - I

- Duplication of nodes

Transformation strategies - I

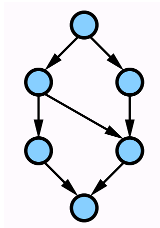
- Duplication of nodes
  - Duplication of nodes related to a forbidden subgraph:

    Reduction sequences [Bein.et al92], path expressions [Naumann94]

Transformation strategies - I

- Duplication of nodes
  - Duplication of nodes related to a forbidden subgraph:

    Reduction sequences [Bein.et al92], path expressions [Naumann94]

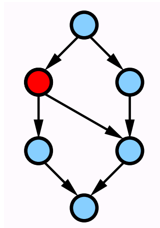


Transformation strategies - I

Theoretical

- Duplication of nodes
  - Duplication of nodes related to a forbidden subgraph:

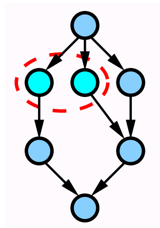
    Reduction sequences [Bein.et al92], path expressions [Naumann94]



Transformation strategies - I

- Duplication of nodes
  - Duplication of nodes related to a forbidden subgraph:

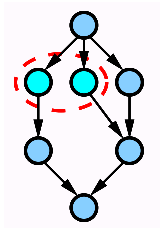
    Reduction sequences [Bein.et al92], path expressions [Naumann94]



Transformation strategies - I

- Duplication of nodes
  - Duplication of nodes related to a forbidden subgraph:

    Reduction sequences [Bein.et al92], path expressions [Naumann94]



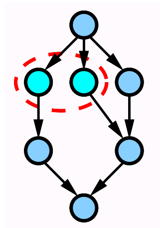
Theoretical

Transformation strategies - I

Theoretical

- Duplication of nodes
  - Duplication of nodes related to a forbidden subgraph:

    Reduction sequences [Bein.et al92], path expressions [Naumann94]



  - Non-work-preserving technique
  - Number of duplications depends on the number of adjacent edges

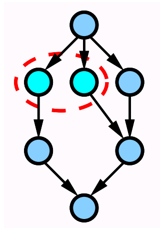
    Increasing number of resources (processing elements) needed

Transformation strategies - I

Theoretical

- Duplication of nodes
  - Duplication of nodes related to a forbidden subgraph:

    Reduction sequences [Bein.et al92], path expressions [Naumann94]



  - Non-work-preserving technique
  - Number of duplications depends on the number of adjacent edges

    Increasing number of resources (processing elements) needed

  - Not appropriate for general purposes

Transformation strategies - II

- Added dependencies

Theoretical

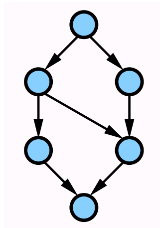
Transformation strategies - II

- Added dependencies
  - Resynchronize parts of the graph related to forbidden subgraphs

Transformation strategies - II

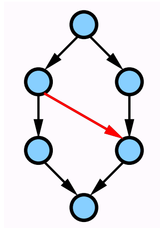
Theoretical

- Added dependencies
  - Resynchronize parts of the graph related to forbidden subgraphs



Transformation strategies - II

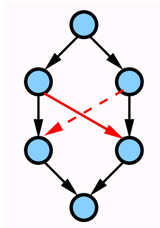
- Added dependencies
  - Resynchronize parts of the graph related to forbidden subgraphs



Theoretical

Transformation strategies - II

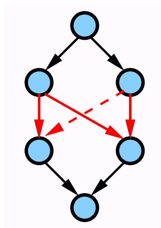
- Added dependencies
  - Resynchronize parts of the graph related to forbidden subgraphs



Theoretical

Transformation strategies - II

- Added dependencies
  - Resynchronize parts of the graph related to forbidden subgraphs

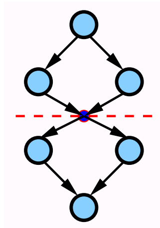


Theoretical

Transformation strategies - II

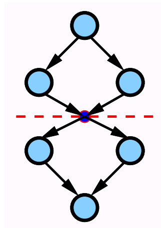
Theoretical

- Added dependencies
  - Resynchronize parts of the graph related to forbidden subgraphs



Transformation strategies - II

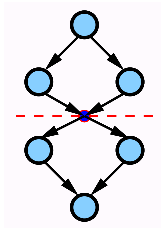
- Added dependencies
  - Resynchronize parts of the graph related to forbidden subgraphs



Theoretical

Transformation strategies - II

- Added dependencies
  - Resynchronize parts of the graph related to forbidden subgraphs

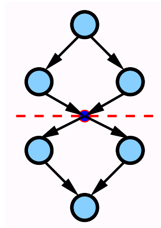


  - Work-preserving technique

Theoretical

Transformation strategies - II

- Added dependencies
  - Resynchronize parts of the graph related to forbidden subgraphs

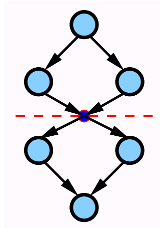


  - Work-preserving technique
  - Serialize potentially parallel tasks: Possible performance loss

Theoretical

Transformation strategies - II

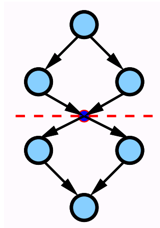
- Added dependencies
  - Resynchronize parts of the graph related to forbidden subgraphs



  - Work-preserving technique
  - Serialize potentially parallel tasks: Possible performance loss
  - We name these techniques as SP-izations

Transformation strategies - II

- Added dependencies
  - Resynchronize parts of the graph related to forbidden subgraphs

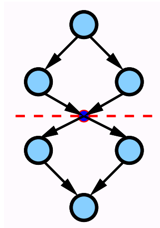


  - Work-preserving technique
  - Serialize potentially parallel tasks: Possible performance loss
  - We name these techniques as SP-izations

- Mixed techniques: Use both strategies

Transformation strategies - II

- Added dependencies
  - Resynchronize parts of the graph related to forbidden subgraphs



  - Work-preserving technique
  - Serialize potentially parallel tasks: Possible performance loss
  - We name these techniques as SP-izations

- Mixed techniques: Use both strategies

- We focus on SP-izations

Layering technique

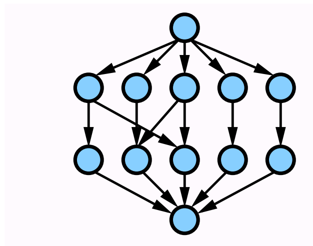
- Well-known system [Malony.et al94] associated to the bulk-synchronous concept

Layering technique

- Well-known system [Malony.et al94] associated to the bulk-synchronous concept
- Procedure:
  1. Compute depth level of any node (layers)
  2. Resynchronize with full barrier between consecutive layers

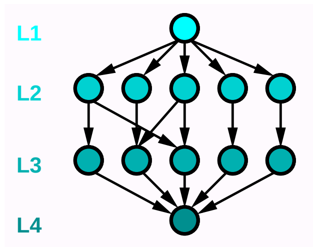
Layering technique

- Well-known system [Malony.et al94] associated to the bulk-synchronous concept
- Procedure:
  1. Compute depth level of any node (layers)
  2. Resynchronize with full barrier between consecutive layers



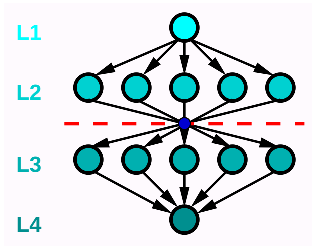
Layering technique

- Well-known system [Malony.et al94] associated to the bulk-synchronous concept
- Procedure:
  1. Compute depth level of any node (layers)
  2. Resynchronize with full barrier between consecutive layers



Layering technique

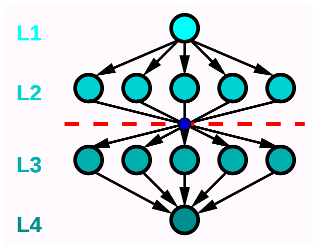
- Well-known system [Malony.et al94] associated to the bulk-synchronous concept
- Procedure:
  1. Compute depth level of any node (layers)
  2. Resynchronize with full barrier between consecutive layers



Layering technique

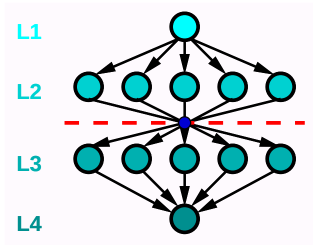
Theoretical

- Well-known system [Malony.et al94] associated to the bulk-synchronous concept
- Procedure:
    1. Compute depth level of any node (layers)
    2. Resynchronize with full barrier between consecutive layers



Layering technique

- Well-known system [Malony.et al94] associated to the bulk-synchronous concept
- Procedure:
  1. Compute depth level of any node (layers)
  2. Resynchronize with full barrier between consecutive layers

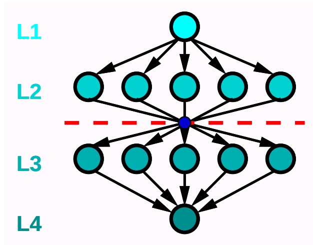


- Low complexity bounds: $O(m + n)$

Layering technique

Theoretical

- Well-known system [Malony.et al94] associated to the bulk-synchronous concept
- Procedure:
  1. Compute depth level of any node (layers)
  2. Resynchronize with full barrier between consecutive layers



- Low complexity bounds: $O(m+n)$
- It does not exploit SP graphs or possibility of local resynchronizations

Algorithm 1

- Local problems solving + Keep SP subgraphs untouched

Theoretical

Algorithm 1

Theoretical

- Local problems solving + Keep SP subgraphs untouched
- Procedure:

  1. Reduce SP subgraphs

     Rest of nodes are related to forbidden subgraphs

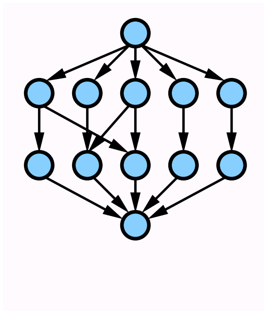
  2. Choose an initial node
  3. Recursive exploration of related nodes
  4. Resynchronization of the NSP problem (two local strategies)
  5. If the graph is not SP goto 1

Algorithm 1

Theoretical

- Local problems solving + Keep SP subgraphs untouched
- Procedure:
  1. Reduce SP subgraphs

     Rest of nodes are related to forbidden subgraphs
  2. Choose an initial node
  3. Recursive exploration of related nodes
  4. Resynchronization of the NSP problem (two local strategies)
  5. If the graph is not SP goto 1

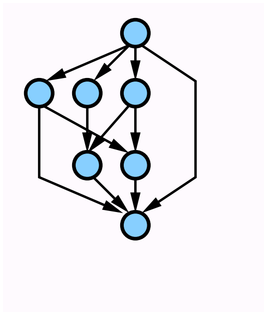


Algorithm 1

- Local problems solving + Keep SP subgraphs untouched
- Procedure:

1. Reduce SP subgraphs

   Rest of nodes are related to forbidden subgraphs
2. Choose an initial node
3. Recursive exploration of related nodes
4. Resynchronization of the NSP problem (two local strategies)
5. If the graph is not SP goto 1



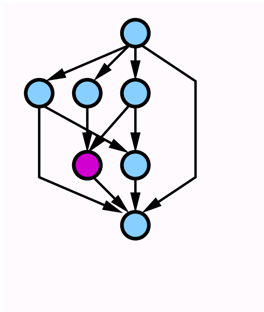
Algorithm 1

- Local problems solving + Keep SP subgraphs untouched
- Procedure:

1. Reduce SP subgraphs

   Rest of nodes are related to forbidden subgraphs

2. Choose an initial node
3. Recursive exploration of related nodes
4. Resynchronization of the NSP problem (two local strategies)
5. If the graph is not SP goto 1



Algorithm 1

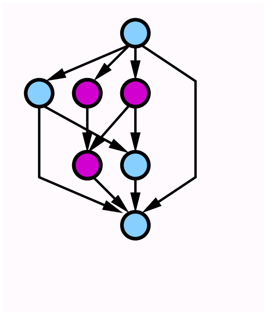
Theoretical

- Local problems solving + Keep SP subgraphs untouched
- Procedure:

1. Reduce SP subgraphs

   Rest of nodes are related to forbidden subgraphs

2. Choose an initial node
3. Recursive exploration of related nodes
4. Resynchronization of the NSP problem (two local strategies)
5. If the graph is not SP goto 1



Algorithm 1

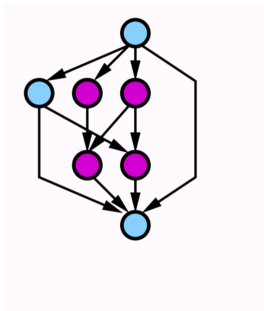
Theoretical

- Local problems solving + Keep SP subgraphs untouched
- Procedure:

  1. Reduce SP subgraphs

     Rest of nodes are related to forbidden subgraphs

  2. Choose an initial node
  3. Recursive exploration of related nodes
  4. Resynchronization of the NSP problem (two local strategies)
  5. If the graph is not SP goto 1



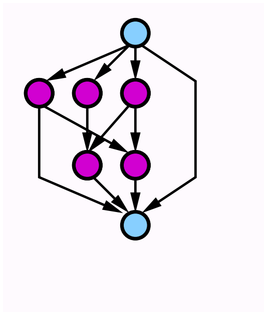
Algorithm 1

- Local problems solving + Keep SP subgraphs untouched
- Procedure:

1. Reduce SP subgraphs

   Rest of nodes are related to forbidden subgraphs

2. Choose an initial node
3. Recursive exploration of related nodes
4. Resynchronization of the NSP problem (two local strategies)
5. If the graph is not SP goto 1



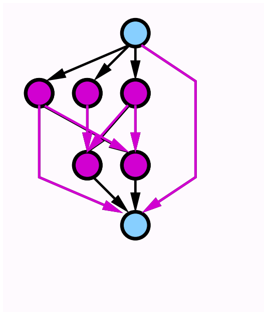
Algorithm 1

- Local problems solving + Keep SP subgraphs untouched
- Procedure:

  1. Reduce SP subgraphs

     Rest of nodes are related to forbidden subgraphs

  2. Choose an initial node
  3. Recursive exploration of related nodes
  4. Resynchronization of the NSP problem (two local strategies)
  5. If the graph is not SP goto 1



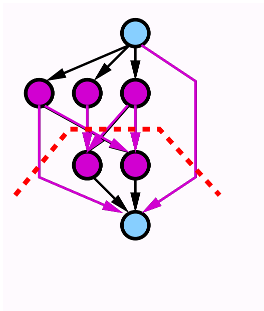
Algorithm 1

- Local problems solving + Keep SP subgraphs untouched
- Procedure:

  1. Reduce SP subgraphs

     Rest of nodes are related to forbidden subgraphs

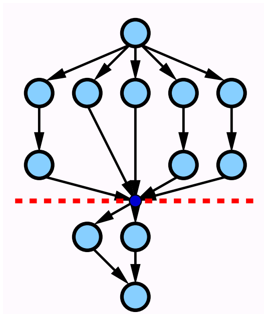
  2. Choose an initial node
  3. Recursive exploration of related nodes
  4. Resynchronization of the NSP problem (two local strategies)
  5. If the graph is not SP goto 1



Algorithm 1

- Local problems solving + Keep SP subgraphs untouched
- Procedure:
  1. Reduce SP subgraphs

     Rest of nodes are related to forbidden subgraphs
  2. Choose an initial node
  3. Recursive exploration of related nodes
  4. Resynchronization of the NSP problem (two local strategies)
  5. If the graph is not SP goto 1



Theoretical

Algorithm 1: Properties

- A local combination is resynchronized in each iteration

Algorithm 1: Properties

- A local combination is resynchronized in each iteration
- No global information stored: $O(m \times n)$

Algorithm 1: Properties

- A local combination is resynchronized in each iteration

- No global information stored: $O(m \times n)$

- It does not keep the layering structure:

  Higher potential overhead even on well-balanced computations

Algorithm 2

- Local resynchronization + Keep layering structure

Theoretical

Algorithm 2

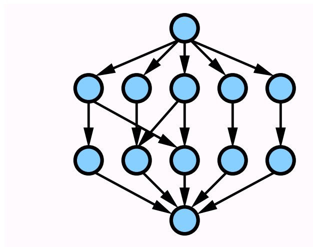
Theoretical

- Local resynchronization + Keep layering structure
- Procedure:
  1. Compute depth levels
  2. For each layer top-down
     (a) Detect local NSP problems between nodes in this and previous layer
     (b) For each problem in any order:
        – Search for the nearest common ancestor of all nodes in the problem
     (c) Recombine ancestors in the minimum set of independent ancestors (use information on the SP tree)
     (d) Merge problems with dependent ancestors
     (e) Separate barrier synchronization for every independent remaining problem

Algorithm 2

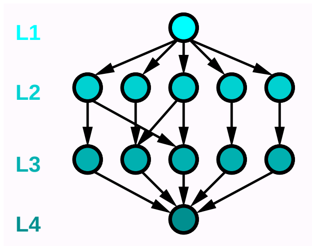
Theoretical

- Local resynchronization + Keep layering structure
- Procedure:
  1. Compute depth levels
  2. For each layer top-down
     (a) Detect local NSP problems between nodes in this and previous layer
     (b) For each problem in any order:
       – Search for the nearest common ancestor of all nodes in the problem
     (c) Recombine ancestors in the minimum set of independent ancestors (use information on the SP tree)
     (d) Merge problems with dependent ancestors
     (e) Separate barrier synchronization for every independent remaining problem



Algorithm 2

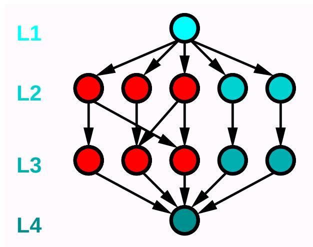
- Local resynchronization + Keep layering structure
- Procedure:
  1. Compute depth levels
  2. For each layer top-down
     (a) Detect local NSP problems between nodes in this and previous layer
     (b) For each problem in any order:
       – Search for the nearest common ancestor of all nodes in the problem
     (c) Recombine ancestors in the minimum set of independent ancestors (use information on the SP tree)
     (d) Merge problems with dependent ancestors
     (e) Separate barrier synchronization for every independent remaining problem



Algorithm 2

Theoretical

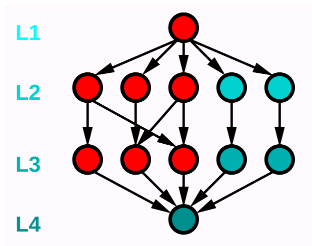
- Local resynchronization + Keep layering structure
- Procedure:
  1. Compute depth levels
  2. For each layer top-down
     (a) Detect local NSP problems between nodes in this and previous layer
     (b) For each problem in any order:
       – Search for the nearest common ancestor of all nodes in the problem
     (c) Recombine ancestors in the minimum set of independent ancestors (use information on the SP tree)
     (d) Merge problems with dependent ancestors
     (e) Separate barrier synchronization for every independent remaining problem



Algorithm 2

Theoretical

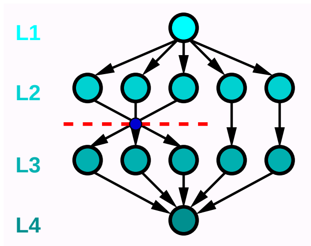
- Local resynchronization + Keep layering structure
- Procedure:
    1. Compute depth levels
    2. For each layer top-down
        - (a) Detect local NSP problems between nodes in this and previous layer
        - (b) For each problem in any order:
            - Search for the nearest common ancestor of all nodes in the problem
        - (c) Recombine ancestors in the minimum set of independent ancestors (use information on the SP tree)
        - (d) Merge problems with dependent ancestors
        - (e) Separate barrier synchronization for every independent remaining problem



Algorithm 2

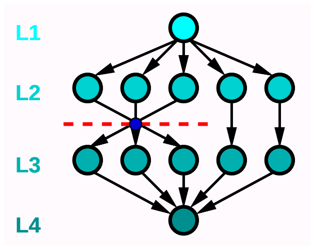
Theoretical

- Local resynchronization + Keep layering structure
- Procedure:
  1. Compute depth levels
  2. For each layer top-down
     (a) Detect local NSP problems between nodes in this and previous layer
     (b) For each problem in any order:
        – Search for the nearest common ancestor of all nodes in the problem
     (c) Recombine ancestors in the minimum set of independent ancestors (use information on the SP tree)
     (d) Merge problems with dependent ancestors
     (e) Separate barrier synchronization for every independent remaining problem



Algorithm 2

- Local resynchronization + Keep layering structure
- Procedure:
  1. Compute depth levels
  2. For each layer top-down
     (a) Detect local NSP problems between nodes in this and previous layer
     (b) For each problem in any order:
       – Search for the nearest common ancestor of all nodes in the problem
     (c) Recombine ancestors in the minimum set of independent ancestors (use information on the SP tree)
     (d) Merge problems with dependent ancestors
     (e) Separate barrier synchronization for every independent remaining problem



Algorithm 2: Properties

- Considers global information stored in the SP tree-reduction

Algorithm 2: Properties

- Considers global information stored in the SP tree-reduction
- Tight time complexity bounds: $O(m + n \log n)$

  Higher complexity than the layering technique

Algorithm 2: Properties

- Considers global information stored in the SP tree-reduction
- Tight time complexity bounds: $O(m + n \log n)$

  Higher complexity than the layering technique
- Similar results as layering for regular NSP structures

  But better results for more irregular, or closer to SP form graphs

Impact indicator

- Objective: Measure the potential performance impact of an SP-ization

  Critical path value (cpv) analysis: Cost model of performance

Theoretical

Impact indicator

- Objective: Measure the potential performance impact of an SP-ization

  Critical path value (cpv) analysis: Cost model of performance

- Relative critical path difference:

$$\gamma_{\tau}(G, G') = \frac{cpv(G')}{cpv(G)}$$

Theoretical

Impact indicator

Theoretical

- Objective: Measure the potential performance impact of an SP-ization

  Critical path value (cpv) analysis: Cost model of performance

- Relative critical path difference:

$$\gamma_\tau(G, G') = \frac{cpv(G')}{cpv(G)}$$

- Upper bounds:
  - Unlikely cases of highly unbalanced computations

    Pathological workload distributions
  - Average cost is more appropriate for dynamic workloads

    [LamportLynch90]

Impact indicator

- Objective: Measure the potential performance impact of an SP-ization

  Critical path value (cpv) analysis: Cost model of performance

- Relative critical path difference:

$$\gamma_{\tau}(G, G') = \frac{cpv(G')}{cpv(G)}$$

- Upper bounds:
  - Unlikely cases of highly unbalanced computations

    Pathological workload distributions
  - Average cost is more appropriate for dynamic workloads

    [LamportLynch90]

- We focus on expected values: $\overline{\gamma}$

Theoretical

Impact indicator

Theoretical

- Objective: Measure the potential performance impact of an SP-ization

  Critical path value (cpv) analysis: Cost model of performance

- Relative critical path difference:

$$\gamma_{\tau}(G, G') = \frac{cpv(G')}{cpv(G)}$$

- Upper bounds:
  - Unlikely cases of highly unbalanced computations

    Pathological workload distributions
  - Average cost is more appropriate for dynamic workloads

    [LamportLynch90]

- We focus on expected values: $\overline{\gamma}$

- Other structural impact metrics are not related with the potential performance loss [Dissertation 3.6.2]

SP vs. NSP cpv analysis

- Absence of real workload information: Stochastic workloads

Theoretical

SP vs. NSP cpv analysis

- Absence of real workload information: Stochastic workloads

  Nodes workload are i.i.d. (independent identically distributed) random variables

SP vs. NSP cpv analysis

- Absence of real workload information: Stochastic workloads

  Nodes workload are i.i.d. (independent identically distributed) random variables

- Stochastic SP cpv analysis:
  - Series composition: Addition of i.i.d. random variables
  - Parallel composition: Order statistics [Gumbel62]

SP vs. NSP cpv analysis

Theoretical

- Absence of real workload information: Stochastic workloads

  Nodes workload are i.i.d. (independent identically distributed) random variables

- Stochastic SP cpv analysis:
  - Series composition: Addition of i.i.d. random variables
  - Parallel composition: Order statistics [Gumbel62]

- Stochastic NSP cpv analysis: Prevented by its inherent complexity

SP vs. NSP cpv analysis

- Absence of real workload information: Stochastic workloads

  Nodes workload are i.i.d. (independent identically distributed) random variables

- Stochastic SP cpv analysis:
  - Series composition: Addition of i.i.d. random variables
  - Parallel composition: Order statistics [Gumbel62]

- Stochastic NSP cpv analysis: Prevented by its inherent complexity

- Approximations for simple regular NSP structures [Vaca99]

SP vs. NSP cpv analysis

- Absence of real workload information: Stochastic workloads

  Nodes workload are i.i.d. (independent identically distributed) random variables

- Stochastic SP cpv analysis:
  - Series composition: Addition of i.i.d. random variables
  - Parallel composition: Order statistics [Gumbel62]

- Stochastic NSP cpv analysis: Prevented by its inherent complexity

- Approximations for simple regular NSP structures [Vaca99]
  - Simple analytical formulae for γ
  - Predicts experimental behavior asymptotically (Error < 25%)
  - Topology-dependent results

SP vs. NSP cpv analysis

Theoretical

- Absence of real workload information: Stochastic workloads

  Nodes workload are i.i.d. (independent identically distributed) random variables

- Stochastic SP cpv analysis:
  - Series composition: Addition of i.i.d. random variables
  - Parallel composition: Order statistics [Gumbel62]

- Stochastic NSP cpv analysis: Prevented by its inherent complexity

- Approximations for simple regular NSP structures [Vaca99]
  - Simple analytical formulae for γ
  - Predicts experimental behavior asymptotically (Error $< 25\%$)
  - Topology-dependent results

- Further experimental study is needed

  to predict the loss of performance in a generic case

Experiments purpose

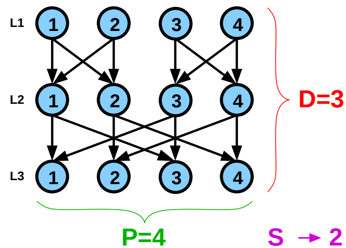
- Objective:
  - Measure the performance loss introduced when NSP structures are programmed in SP form
  - Relate the performance loss to graph or workload parameters

Experiments purpose

Experimental

- Objective:
  - Measure the performance loss introduced when NSP structures are programmed in SP form
  - Relate the performance loss to graph or workload parameters

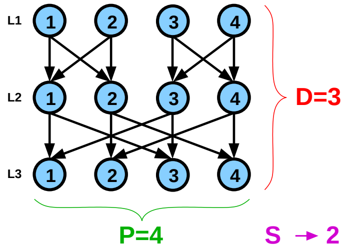
- Topological parameters:
  - P: Maximum degree of parallelism
  - D: Maximum depth level
  - S: Synchronization density



Experiments purpose

- Objective:
  - Measure the performance loss introduced when NSP structures are programmed in SP form
  - Relate the performance loss to graph or workload parameters

- Topological parameters:
  - P: Maximum degree of parallelism
  - D: Maximum depth level
  - S: Synchronization density



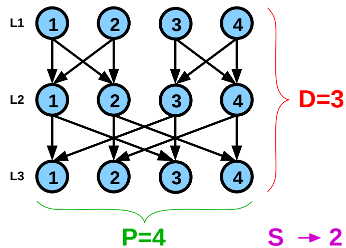
- Workload parameters: $\tau \rightsquigarrow \mathcal{D}(\mu, \sigma)$
  - Relative deviation: $\varsigma = \frac{\sigma}{\mu}$ (”variability”)

Experiments purpose

Experimental

- Objective:
  - Measure the performance loss introduced when NSP structures are programmed in SP form
  - Relate the performance loss to graph or workload parameters

- Topological parameters:
  - P: Maximum degree of parallelism
  - D: Maximum depth level
  - S: Synchronization density



- Workload parameters: $\tau \rightsquigarrow \mathcal{D}(\mu, \sigma)$
  - Relative deviation: $\varsigma = \frac{\sigma}{\mu}$ ("variability")

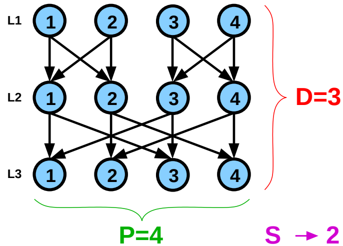
- Sizes: From small ($P, D < 10$) to large ($P, D > 1000$)

Experiments purpose

Experimental

- Objective:
  - Measure the performance loss introduced when NSP structures are programmed in SP form
  - Relate the performance loss to graph or workload parameters

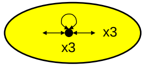
- Topological parameters:
  - P: Maximum degree of parallelism
  - D: Maximum depth level
  - S: Synchronization density



- Workload parameters: $\tau \rightsquigarrow \mathcal{D}(\mu, \sigma)$
  - Relative deviation: $\varsigma = \frac{\sigma}{\mu}$ ("variability")

- Sizes: From small ($P, D < 10$) to large ($P, D > 1000$)
- Variability: From balanced ($\varsigma = 0.1$) to highly unbalanced ($\varsigma = 1$)

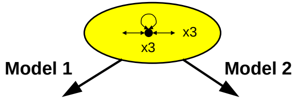
γ levels of detail



Algorithm

Experimental

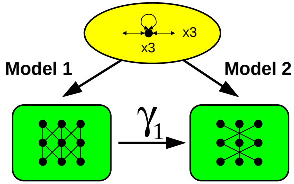
γ levels of detail



Algorithm

Experimental

γ levels of detail

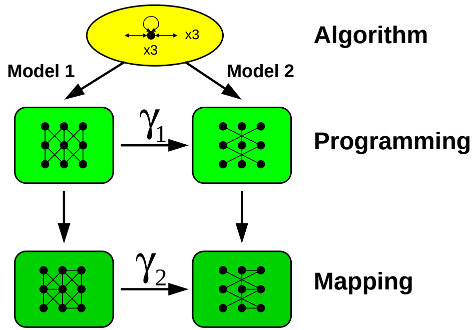


Algorithm

Programming

Experimental

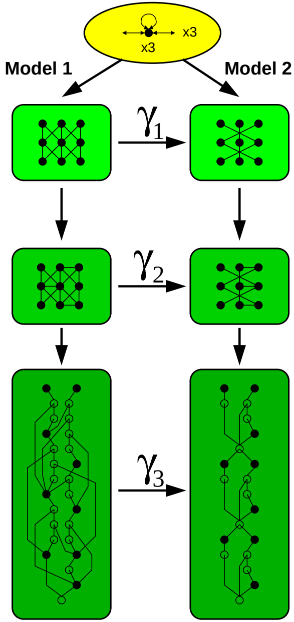
γ levels of detail



Experimental

γ levels of detail

Experimental



Algorithm

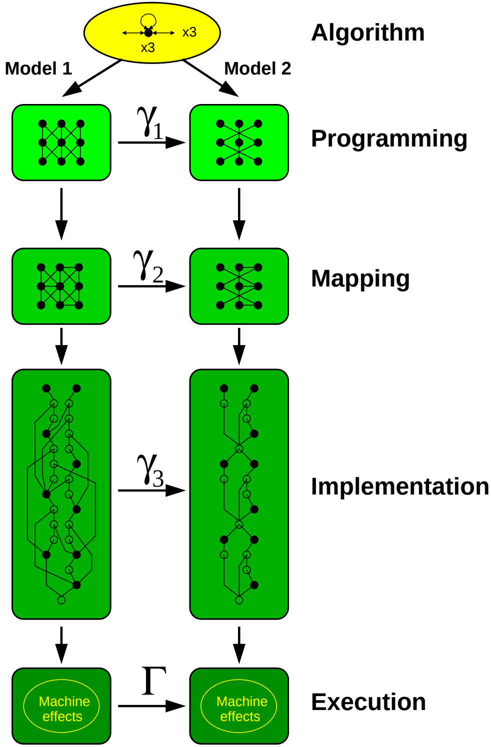
Programming

Mapping

Implementation

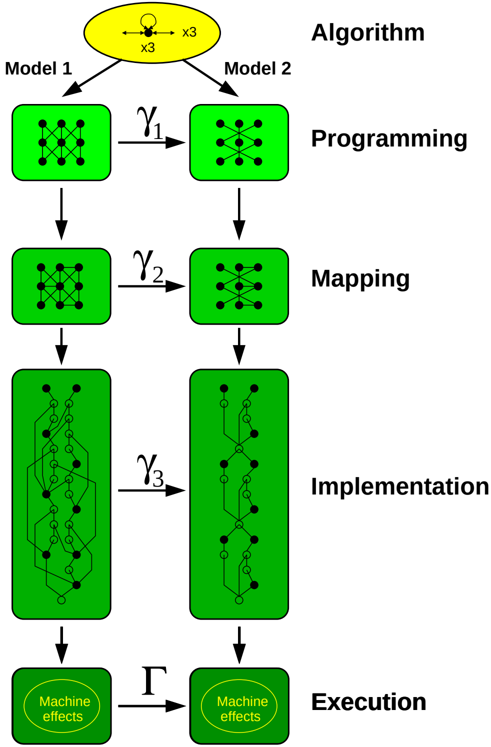
γ levels of detail

Experimental



γ levels of detail

Experimental



Experiments design - I

- Exhaustive testing of the graph space is impossible

Experiments design - I

- Exhaustive testing of the graph space is impossible
- Approaches:

Experiments design - I

- Exhaustive testing of the graph space is impossible
- Approaches:

Experiments $\left\{ \vphantom{\begin{matrix} a \\ b \\ c \\ d \\ e \\ f \end{matrix}} \right.$

Experiments design - I

- Exhaustive testing of the graph space is impossible
- Approaches:

Experiments
- Synthetic graphs
  - Random sample graphs
  - Meshes

Experiments design - I

Experimental

- Exhaustive testing of the graph space is impossible
- Approaches:

Experiments
- Synthetic graphs
  - Random sample graphs
  - Meshes
- Real applications
  - Static applications
  - Dynamic applications

Experiments design - II

Synthetic graphs

- Random sample of the graph space: General idea of trends

Experiments design - II

Experimental

Synthetic graphs

- Random sample of the graph space: General idea of trends
  - Random graphs generation technique [Almeida92]
  - Parameters: Size, S, ς

Experiments design - II

Synthetic graphs

- Random sample of the graph space: General idea of trends
  - Random graphs generation technique [Almeida92]
  - Parameters: Size, S, ς
- Meshes: Regular topologies of i layers with j nodes each

Experiments design - II

Synthetic graphs

- Random sample of the graph space: General idea of trends
  - Random graphs generation technique [Almeida92]
  - Parameters: Size, S, ς
- Meshes: Regular topologies of i layers with j nodes each
  - Regular or random synchronization between consecutive layers [TobitaKasahara99]
  - Parameters: P, D, S, ς

Experiments design - II

Synthetic graphs

- Random sample of the graph space: General idea of trends
  - Random graphs generation technique [Almeida92]
  - Parameters: Size, S, ς
- Meshes: Regular topologies of i layers with j nodes each
  - Regular or random synchronization between consecutive layers [TobitaKasahara99]
  - Parameters: P, D, S, ς
- Workload: 25 draws for each topology and ς value

Experiments design - III

Real static applications

- Easy graph modeling at any level of detail

Experimental

Experiments design - III

Real static applications

- Easy graph modeling at any level of detail
- Typically highly regular: Results expected to be similar than meshes

Experiments design - III

Real static applications

- Easy graph modeling at any level of detail
- Typically highly regular: Results expected to be similar than meshes
- Parameters: P, D

Experiments design - III

Real static applications

- Easy graph modeling at any level of detail
- Typically highly regular: Results expected to be similar than meshes
- Parameters: $\boldsymbol{P}, \boldsymbol{D}$
- Macro-pipeline, Cellular Automata, FFT, LU reduction

Experiments design - III

Experimental

Real static applications

- Easy graph modeling at any level of detail
- Typically highly regular: Results expected to be similar than meshes
- Parameters: P, D
- Macro-pipeline, Cellular Automata, FFT, LU reduction
- Framework (γ, Γ):
  - Programming/mapping levels: Synthetic workloads
  - Implementation level: Communication costs considered
  - Execution level: MPI implementations (SP version with barriers)

[Dissertation 4.2.1]

Experiments design - III

Real static applications

- Easy graph modeling at any level of detail
- Typically highly regular: Results expected to be similar than meshes
- Parameters: P, D
- Macro-pipeline, Cellular Automata, FFT, LU reduction
- Framework (γ, Γ):
  - Programming/mapping levels: Synthetic workloads
  - Implementation level: Communication costs considered
  - Execution level: MPI implementations (SP version with barriers)

    [Dissertation 4.2.1]
- Execution level: Three architectures
  - CC-NUMA (Origin2000)
  - Message-passing with low latency (CrayT3E)
  - Distributed memory with high latency (Beowulf)

Experiments design - IV

Experimental

Real dynamic applications

- Two cases:
  1. Structure can be reconstructed from input data structure
  2. Structure can be obtained only by tracing in run-time

Experiments design - IV

Experimental

Real dynamic applications

- Two cases:
    1. Structure can be reconstructed from input data structure
    2. Structure can be obtained only by tracing in run-time

- Typically more irregular than static applications
- One example application of each type

- Six real input data examples for each application

Experiments design - V

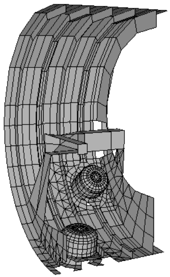
Iterative PDE solver

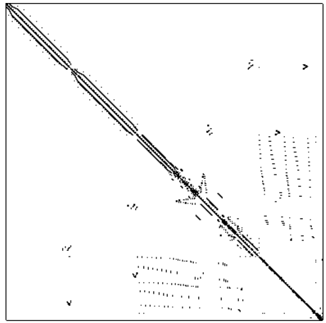
Experiments design - V

Iterative PDE solver

- Six real structural engineering examples

  Matrix Market: Harwell-Boeing, Everstine's collection.



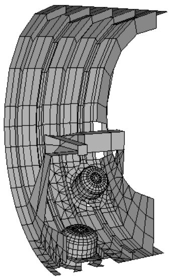


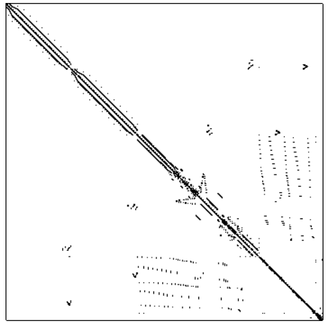
Experiments design - V

Iterative PDE solver

- Six real structural engineering examples

  Matrix Market: Harwell-Boeing, Everstine's collection.





- Sparse matrix data is partitioned for data-layout

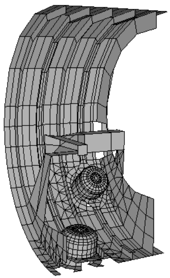
  State-of-the-art partitioning software: METIS

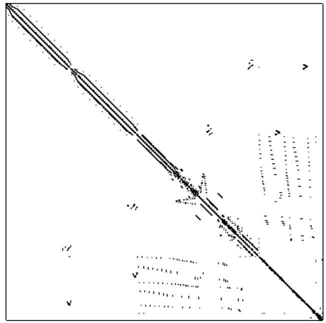
Experiments design - V

Iterative PDE solver

- Six real structural engineering examples

  Matrix Market: Harwell-Boeing, Everstine's collection.





- Sparse matrix data is partitioned for data-layout

  State-of-the-art partitioning software: METIS

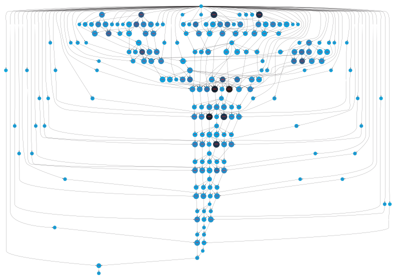
- Mapping level graph reconstructed
- Workload per task estimated as a function of data-layout

Experiments design - VI

Experimental

Domain decomposition and sparse matrix factorization

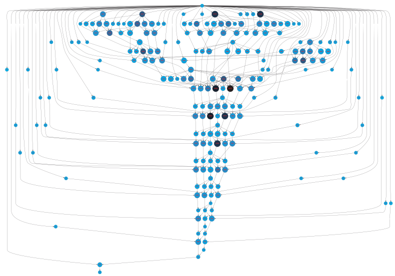
- Real software oriented to structural engineering: DIANA + Tgex



Experiments design - VI

Domain decomposition and sparse matrix factorization

- Real software oriented to structural engineering: DIANA + Tgex

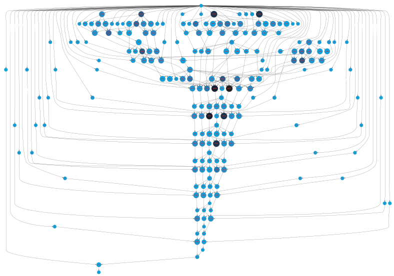


- We use six example mapping level graphs reconstructed from tracing information obtained in a previous work [Lin94,96]

Experiments design - VI

Domain decomposition and sparse matrix factorization

- Real software oriented to structural engineering: DIANA + Tgex

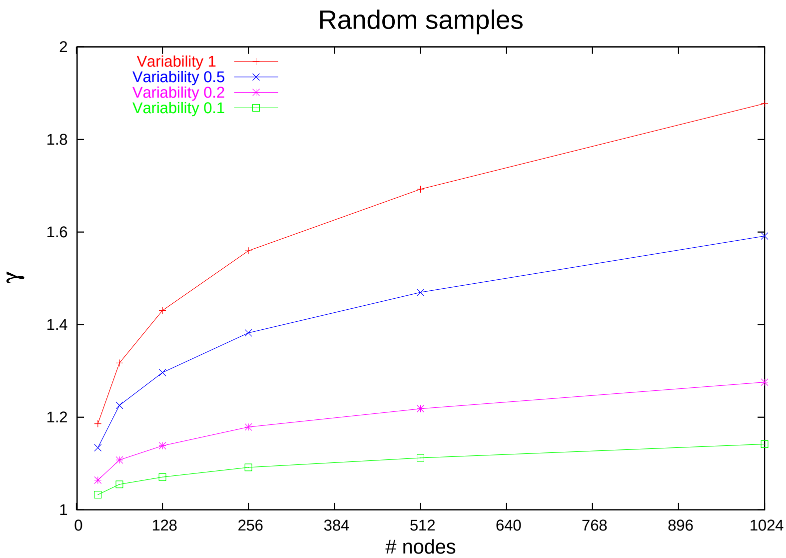


- We use six example mapping level graphs reconstructed from tracing information obtained in a previous work [Lin94,96]

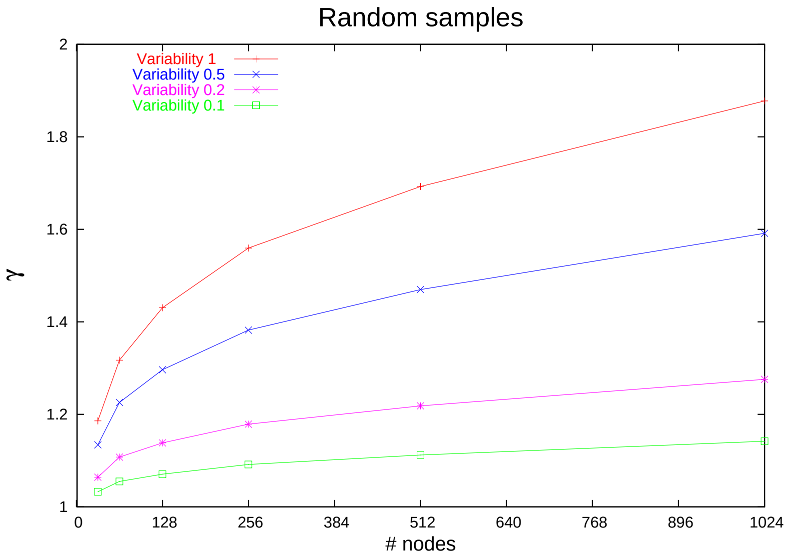
- Real execution workloads provided

Results: Workload

Experimental

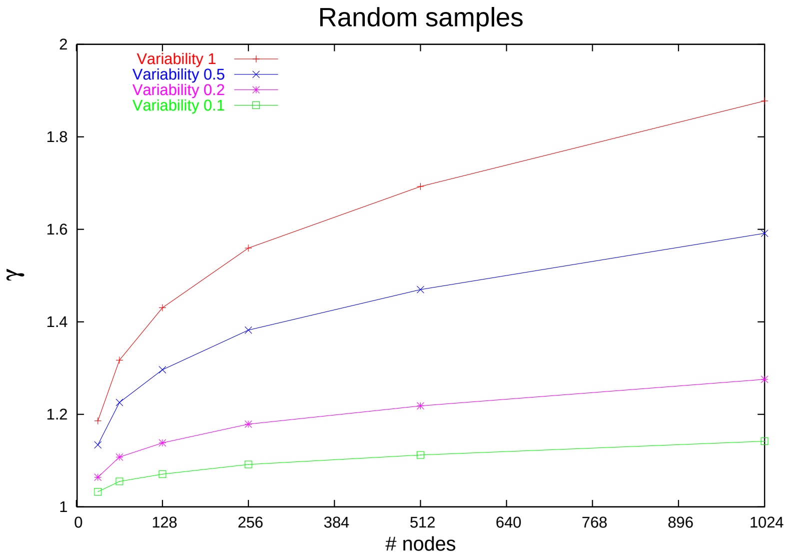


Results: Workload



- Low workload unbalance ⟶ Minimal performance loss
- High workload unbalance ⟶ Increasing performance loss

Results: Workload



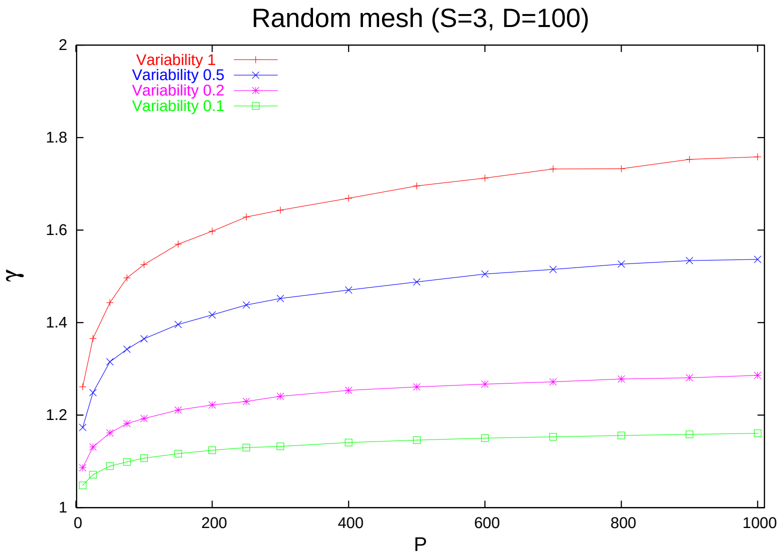
- Low workload unbalance $\longrightarrow$ Minimal performance loss
- High workload unbalance $\longrightarrow$ Increasing performance loss
- Workload correlation with layers or vertical instances of nodes

  Reduced performance loss [Dissertation 4.1.3]

Experimental

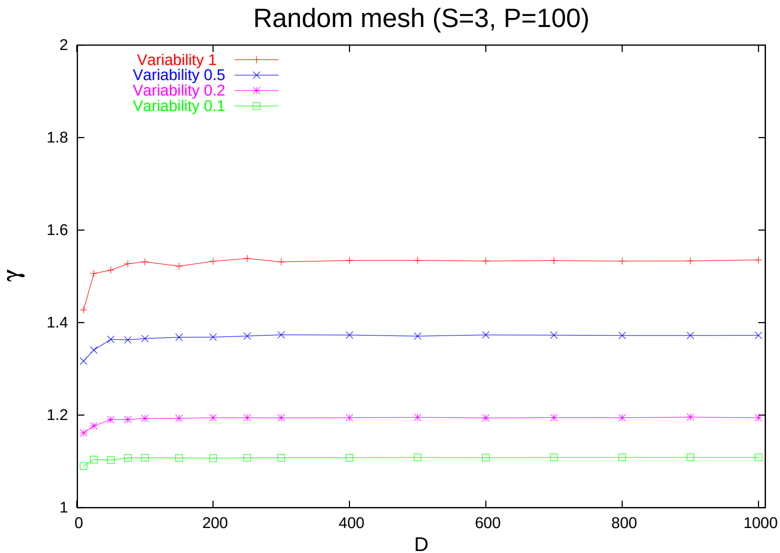
Results: Graph size parameters P, D

Results: Graph size parameters P, D



- P responsible for the under-logarithmic-like loss of performance

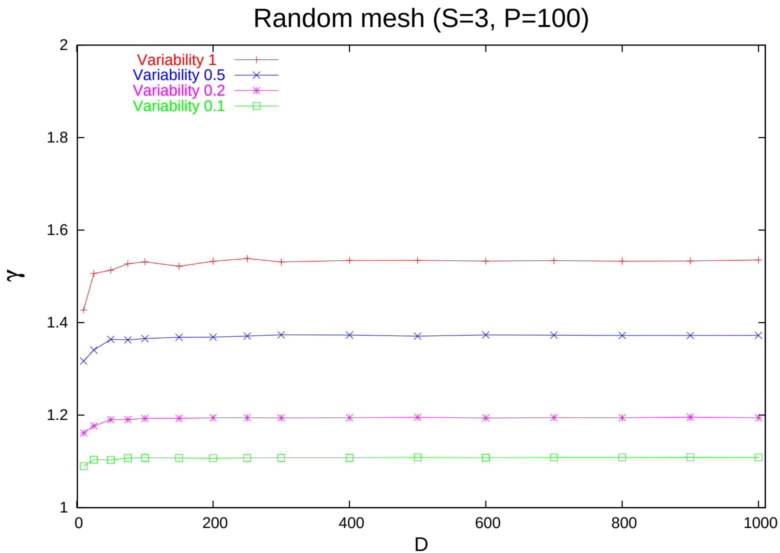
Results: Graph size parameters P, D



- P responsible for the under-logarithmic-like loss of performance
- D has a limited effect

Experimental

Results: Graph size parameters P, D

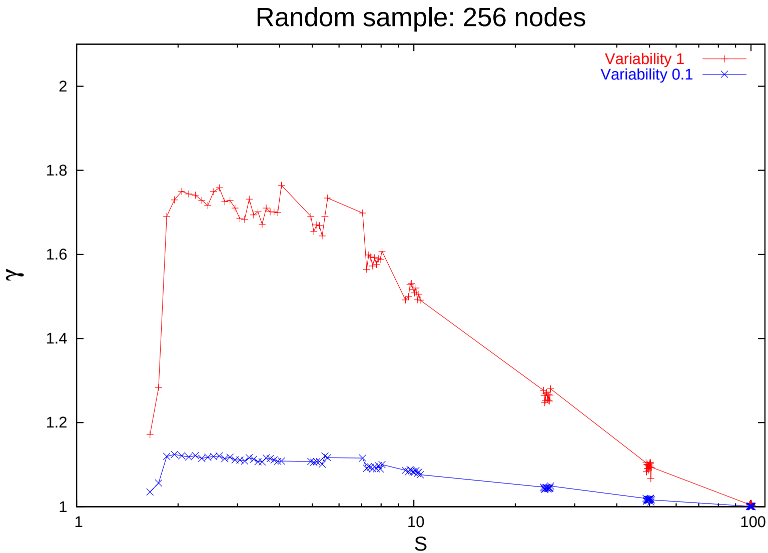


- P responsible for the under-logarithmic-like loss of performance
- D has a limited effect

Pathological effects characterization and metric [Dissertation 4.1.3]

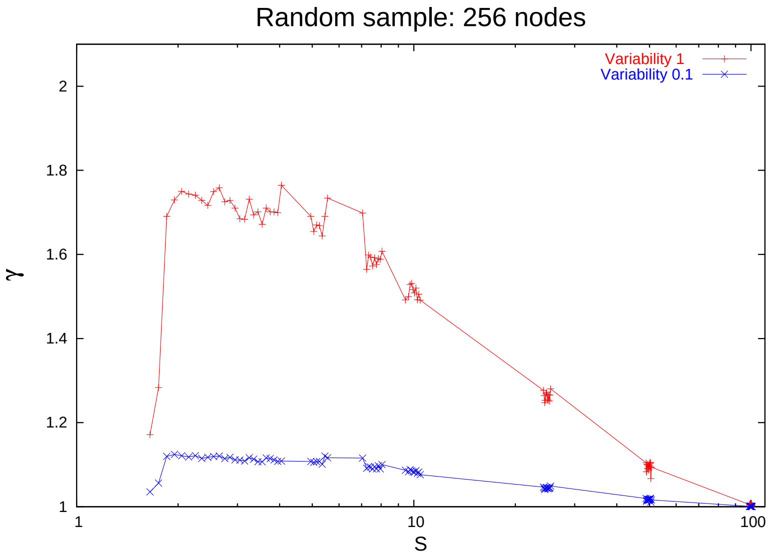
Experimental

Results: Graph parameter S



- S increase has opposite effect to P

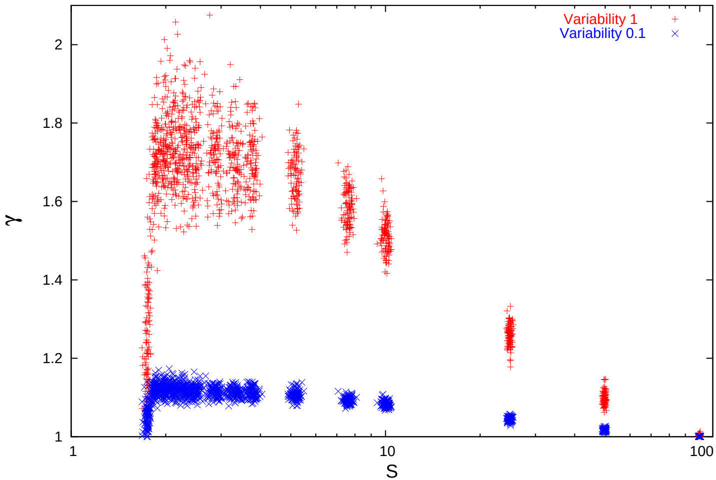
Results: Graph parameter S



- S increase has opposite effect to P
- $S < 2$ implies sparse graphs containing SP series subgraphs

Results: Graph parameter S

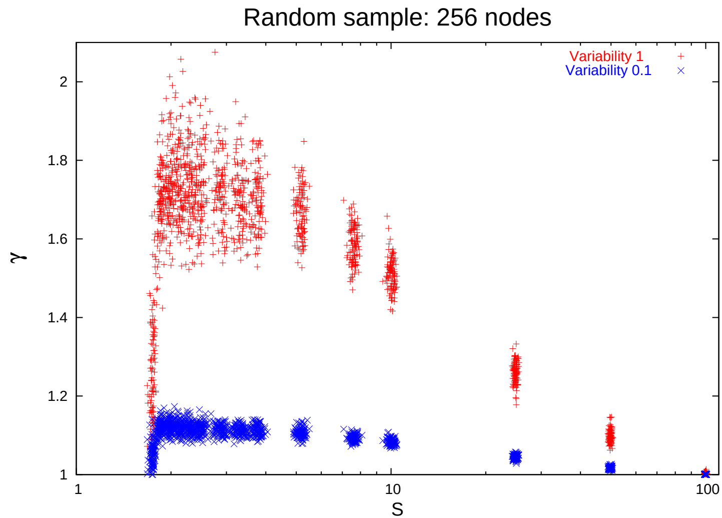
Random sample: 256 nodes



- Maximum dispersion around $S = 2$

Experimental

Results: Graph parameter S



- Maximum dispersion around $S = 2$
- Asymptotic predictions:

$$\overline{\gamma} \approx \frac{\mu + \sigma\sqrt{\log(P)}}{\mu + \sigma\sqrt{\log(S)}}$$

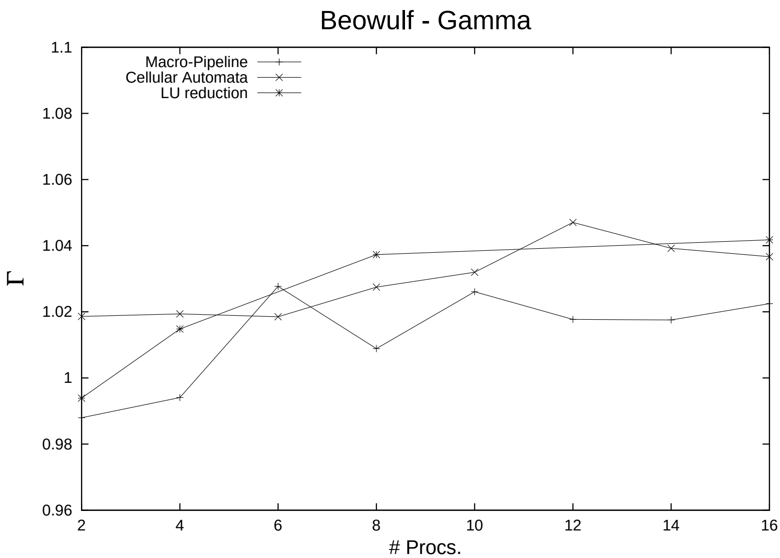
Experimental

Results: Execution level Γ

Experimental

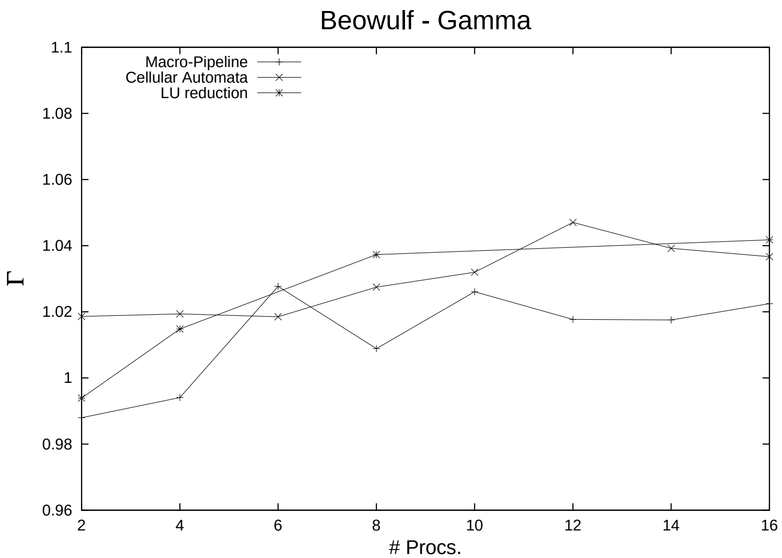
Results: Execution level Γ

Experimental



- Static applications: Extremely balanced workloads, negligible γ

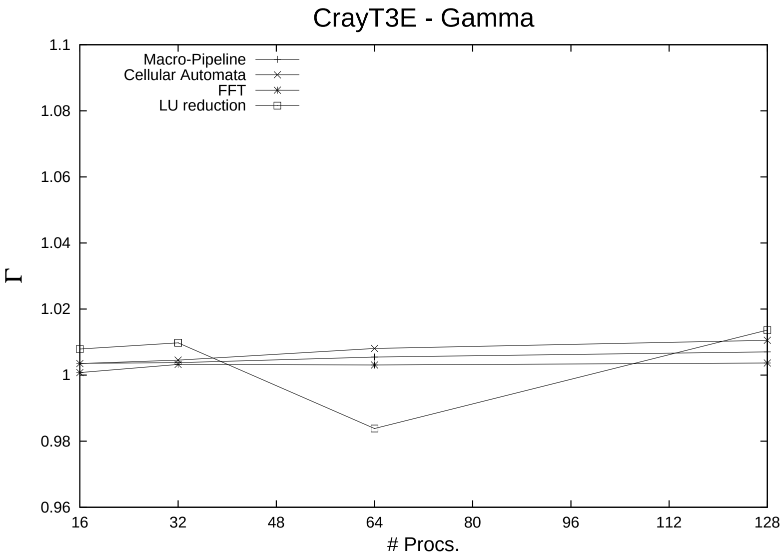
Results: Execution level Γ



- Static applications: Extremely balanced workloads, negligible γ
- Non-optimized communications: Barriers noticeable

Results: Execution level Γ

Experimental



- Static applications: Extremely balanced workloads, negligible γ
- Non-optimized communications: Barriers noticeable

However, sometimes communications perform better in presence of a barrier!

Results: Dynamic applications - I

Sparse iterative solvers

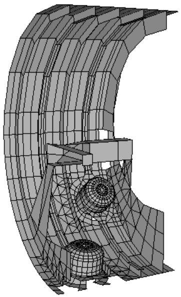
- METIS partitioning produces very well workload and synchronization balance

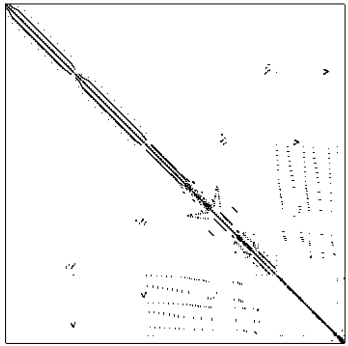
Results: Dynamic applications - I

Experimental

Sparse iterative solvers

- METIS partitioning produces very well workload and synchronization balance



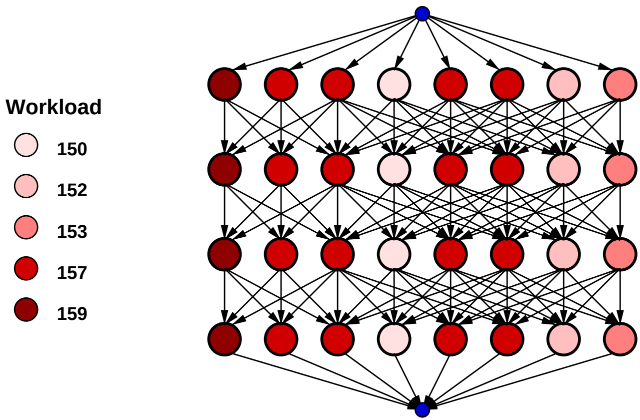


Results: Dynamic applications - I

Experimental

Sparse iterative solvers

- METIS partitioning produces very well workload and synchronization balance

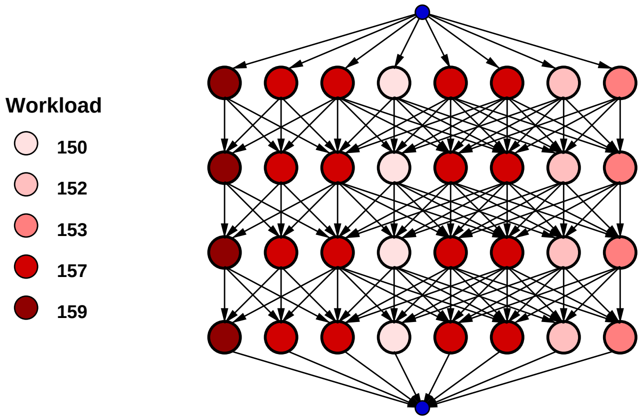


Results: Dynamic applications - I

Experimental

Sparse iterative solvers

- METIS partitioning produces very well workload and synchronization balance



- Negligible loss of performance: Expected for any good load-balancing technique

Results: Dynamic applications - II

Experimental

Domain decomposition and sparse matrix factorization

Results: Dynamic applications - II

Domain decomposition and sparse matrix factorization

- Bad statistical workload parameters: $\varsigma \gg 1$ in most cases

Experimental

Results: Dynamic applications - II

Experimental

Domain decomposition and sparse matrix factorization

- Bad statistical workload parameters: $\varsigma \gg 1$ in most cases
- Experiments with synthetic workloads show lower γ than expected
- Real workload even lower:

# nodes	ς	Γ
59	2.1	1.000
113	3.0	1.006
213	1.4	1.074
528	2.0	1.199
773	7.1	1.009
2015	2.6	1.103

Results: Dynamic applications - II

Experimental

Domain decomposition and sparse matrix factorization

- Bad statistical workload parameters: $\varsigma \gg 1$ in most cases
- Experiments with synthetic workloads show lower γ than expected
- Real workload even lower:

# nodes	ς	Γ
59	2.1	1.000
113	3.0	1.006
213	1.4	1.074
528	2.0	1.199
773	7.1	1.009
2015	2.6	1.103

- Domain decomposition data-layout produces workload and topology regularities

Summary

- Parallel programming field: Lack of a common development direction

Summary

- Parallel programming field: Lack of a common development direction

- We have proposed a new classification system for PPMs, based on SA

  The adequacy of a model in terms of expressive power, software development methodologies and analyzability characteristics, is related to its SA class

Summary

- Parallel programming field: Lack of a common development direction

- We have proposed a new classification system for PPMs, based on SA

  The adequacy of a model in terms of expressive power, software development methodologies and analyzability characteristics, is related to its SA class

- The SP-restriction is a critical decision for a PPM adequacy

  SA: Key for the expressive power vs. analyzability trade-off

Summary

- Parallel programming field: Lack of a common development direction
- We have proposed a new classification system for PPMs, based on SA

  The adequacy of a model in terms of expressive power, software development methodologies and analyzability characteristics, is related to its SA class
- The SP-restriction is a critical decision for a PPM adequacy

  SA: Key for the expressive power vs. analyzability trade-off
- The expressive power restriction associated with SP PPMs has been investigated in-depth both theoretically and empirically

Methodology

- Methodology: Three-way approach
  - Conceptual: Models and applications review, SA classification.

    Qualitative study

Methodology

- Methodology: Three-way approach
  - Conceptual: Models and applications review, SA classification.

    Qualitative study

  - Theoretical: SP, NSP graph characterization
    and algorithmic transformation techniques

Methodology

- Methodology: Three-way approach
  - Conceptual: Models and applications review, SA classification.

    Qualitative study

  - Theoretical: SP, NSP graph characterization and algorithmic transformation techniques
  - Experimental: Empirical analysis framework for the potential negative performance impact of SP programming at different levels of detail, including propagation to execution level

Results

- At the design or programming level (γ):
  Correlation between the SP potential loss of parallelism with simple application parameters:
  - $\boldsymbol{P}$ has an under-logarithmic-like effect on $\boldsymbol{\gamma}$
  - $\boldsymbol{S}$ has a positive inverse effect
  - Variability ($\boldsymbol{\varsigma}$) has the major impact on $\boldsymbol{\gamma}$

Results

- At the design or programming level (γ):

  Correlation between the SP potential loss of parallelism with simple application parameters:

  - $\boldsymbol{P}$ has an under-logarithmic-like effect on $\boldsymbol{\gamma}$
  - $\boldsymbol{S}$ has a positive inverse effect
  - Variability $(\boldsymbol{\varsigma})$ has the major impact on $\boldsymbol{\gamma}$

- At execution level (Γ):

  In our experiments with real applications $\boldsymbol{\Gamma}$ is bounded to tens of per-cents

  It almost does not scale with the problem size!

Results

- At the design or programming level (γ):

  Correlation between the SP potential loss of parallelism with simple application parameters:

  - P has an under-logarithmic-like effect on γ
  - S has a positive inverse effect
  - Variability (ς) has the major impact on γ

- At execution level (Γ):

  In our experiments with real applications Γ is bounded to tens of percents

  It almost does not scale with the problem size!

- SP performance degradation is mainly associated to poorly balanced and unstructured computations

Results

- At the design or programming level (γ):

  Correlation between the SP potential loss of parallelism with simple application parameters:

  - $\boldsymbol{P}$ has an under-logarithmic-like effect on $\boldsymbol{\gamma}$
  - $\boldsymbol{S}$ has a positive inverse effect
  - Variability (ς) has the major impact on $\boldsymbol{\gamma}$

- At execution level (Γ):

  In our experiments with real applications Γ is bounded to tens of percents

  It almost does not scale with the problem size!

- SP performance degradation is mainly associated to poorly balanced and unstructured computations

- SP SA is a promising design concept for portable, efficient, easy-to-use and general-purpose PPMs

On-going and future research

- Further experiments with more irregular applications

On-going and future research

- Further experiments with more irregular applications
- New NSP to SP transformations:
  - Based on both strategies
  - Using information of estimated workload

On-going and future research

- Further experiments with more irregular applications
- New NSP to SP transformations:
  - Based on both strategies
  - Using information of estimated workload
- Real SP programming framework development:

  Automatic mapping and scheduling guided by performance cost analysis

Main contributions

- CPC 2003, Tenth International Workshop on Compilers for Parallel Computers, Amsterdam, The Netherlands
- VecPar 2002, 5th International Meeting, High Performance Computing for Computational Science, Porto, Portugal (Best Student Paper Award)
- CPC 2001, Ninth International Workshop on Compilers for Parallel Computers, Edinburgh, Scotland
- VecPar 2000, 4th International Meeting on Vector and Parallel Processing, Porto, Portugal
- CPC 2000, Eigth International Workshop on Compilers for Parallel Computers, Aussois, France
- Parallel Computing ParCo'99 Delft, The Netherlands
- Euro-PDS'97, IASTED International Conference, Parallel and Distributed Systems, Barcelona
- ASCI'97, Proceedings of the third annual conference of the Advanced School for Computing and Imaging, Heijen

Questions?

Powered by: LaTeX, Xfig, Gnuplot, PPower4, Acrobat Reader