Scheduling Trees of Malleable Tasks for Sparse Linear Algebra

Abdou Guermouche Loris Marchal Bertrand Simon Frédéric Vivien

ENS Lyon

Nov. 2014

Introduction

Motivation

- Parallel workloads → task graphs (DAGs)
- Multifrontal Cholesky/LU sparse matrix factorization → task trees

Overview of the model

- Malleable tasks: tasks are processed on a variable number of processors
- Speedup: p^{α} with $0 < \alpha \leq 1$
  Example: task T_i is alloted p_i processors → processing time $= L_i / p_i^{\alpha}$
- $p \in \mathbb{R}^+$: non-integer number of processors (time-sharing)

Model advocated by Prasanna and Musicus in [PM96]

Main objective

Minimize the processing time of a malleable task tree graph using:

- Tree parallelism
- Task parallelism

Validation of the malleable task model

Validation criteria

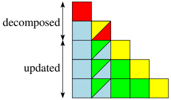
- Relevance of malleable-task tree graphs
- Validation of the speedup model for various shares of processors
- Uniformity of the value of α within an application

Validation of the malleable task model

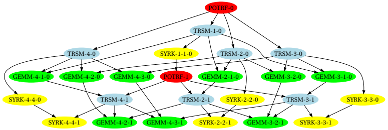
Validation criteria

- Relevance of malleable-task tree graphs
- Validation of the speedup model for various shares of processors
- Uniformity of the value of α within an application

Multifrontal sparse direct solvers → matrix factorization
⇝ tree-shaped task graph (assembly tree)
A task of the assembly tree → partial factorization
⇝ graph of smaller granularity tasks (kernels)



(a) Tiled dense sub-matrix to be partially decomposed



(b) Corresponding kernel graph

Decomposition of a task of the DAG of a Cholesky decomposition into smaller kernels

Validation of the malleable task model

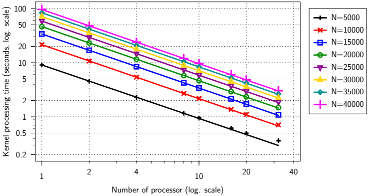
Validation criteria

- Relevance of malleable-task tree graphs
- Validation of the speedup model for various shares of processors
- Uniformity of the value of α within an application

Experiments using the StarPU runtime:

- on 4 10-core processors ($p \leq 40$)
- 3 dense kernels tested: Cholesky, QR (Morse), `qr_mumps`

$\Rightarrow$ Model fits well except for small matrices with a large p



(c) Timings and model for QR on $4096 \times N$ matrices

Validation of the malleable task model

Validation criteria

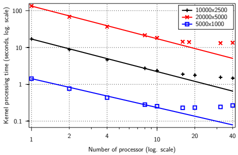
- Relevance of malleable-task tree graphs
- Validation of the speedup model for various shares of processors
- Uniformity of the value of α within an application

Cholesky, QR on dense matrix: $\alpha \approx 1$.

Using `qr_mumps`:

- α: used linear regression on 'small' p
- computed α $\approx$ independent of matrix size
- α depends on
  - Parameters of the problem
  - Memory performance (NUMA)

$\Rightarrow$ Model valid for p smaller than a threshold that should not be reached in practice



(d) Timings and model with 1D partitioning

matrix size	value of α for 1D partitioning	value of α for 2D partitioning
5000×1000	0.78	0.93
10000×2500	0.88	0.95
20000×5000	0.89	0.94

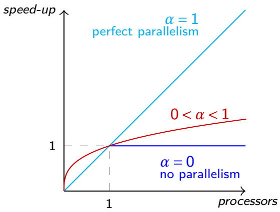
(e) Values of α measured for `qr_mumps` tasks

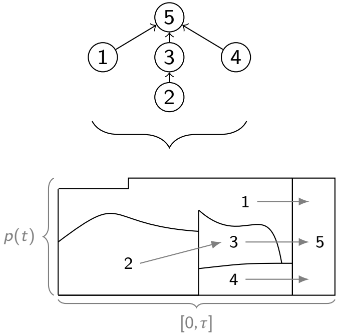
Model and notations

Parameters of the problem

- **In-tree** G of **malleable tasks** of lengths L_i → precedence constraints
- **Speedup** f (= sequential time / parallel time):
  - $f(p) = p^\alpha$ for $0 < \alpha \leq 1$, $p \in \mathbb{R}^+$
  - processing time of T_i: $= \arg\min_C \left\{ \int_0^C p_i(t)^\alpha \, \mathrm{d}t \geq L_i \right\}$
- **Processor profile**: step function $p(t)$, available number of processors at time t

The speedup function





Series-Parallel graphs as a generalization of trees

Motivation

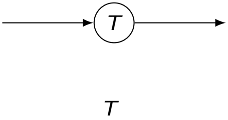
- Our objective: study trees
- Next part: consider more general graphs

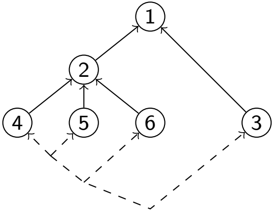
Series-Parallel graphs

Recursively defined by being either:

- a single task
- a parallel composition of two SP graphs
- a series composition of two SP graphs

A tree can be extended to a SP graph.





Series-Parallel graphs as a generalization of trees

Motivation

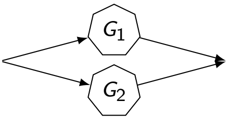
- Our objective: study trees
- Next part: consider more general graphs

Series-Parallel graphs

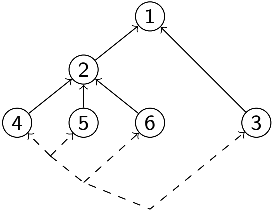
Recursively defined by being either:

- a single task
- a parallel composition of two SP graphs
- a series composition of two SP graphs

A tree can be extended to a SP graph.



$G_1 \parallel G_2$



Series-Parallel graphs as a generalization of trees

Motivation

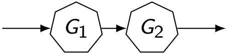
- Our objective: study trees
- Next part: consider more general graphs

Series-Parallel graphs

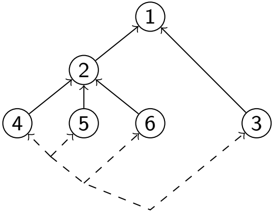
Recursively defined by being either:

- a single task
- a parallel composition of two SP graphs
- a series composition of two SP graphs

A tree can be extended to a SP graph.



$G_1 ; G_2$



Outline

1. Introduction and notations
2. Minimizing the makespan
   - Characterization of the optimal schedule
   - Scheme of the proof of the theorem
3. Extensions to distributed memory
   - Homogeneous multicore nodes
   - Heterogeneous multicore nodes
4. Gain of speedup-aware strategies
5. Conclusion

Outline

1. Introduction and notations
2. Minimizing the makespan
    - Characterization of the optimal schedule
    - Scheme of the proof of the theorem
3. Extensions to distributed memory
4. Gain of speedup-aware strategies
5. Conclusion

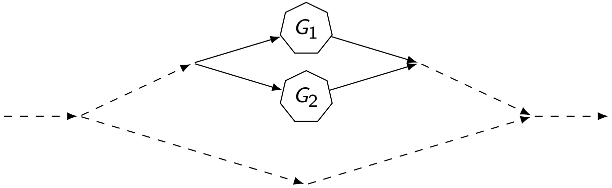
Statement of the problem

Context and hypotheses

- Objects of interest: miminum-makespan schedules of a SP graph G
- [PM96] proved these results using *heavy* optimal control theory
- Our objective: reprove it using pure-scheduling arguments

Theorem (Prasanna & Musicus)

In optimal schedules, at any parallel node $G_1 \parallel G_2$, the ratio of processors given to each branch is constant.



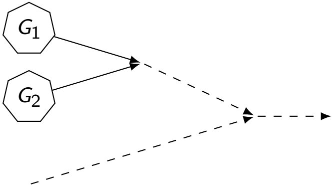
Statement of the problem

Context and hypotheses

- Objects of interest: miminum-makespan schedules of a SP graph G
- [PM96] proved these results using *heavy* optimal control theory
- Our objective: reprove it using pure-scheduling arguments

Theorem (Prasanna & Musicus)

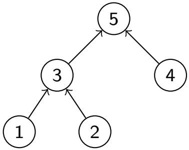
In optimal schedules, at any parallel node $G_1 \parallel G_2$, the ratio of processors given to each branch is constant.

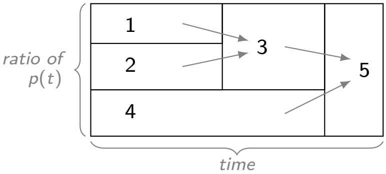


Consequences of the theorem

Corollary

- *In optimal schedules:*
  - $\forall i,\ p_i(t)/p(t)$ *is constant*
  - *Children of a node terminate simultaneously*
- $G \approx$ *equivalent task* T_G *of length* $\mathscr{L}_G$ *defined by:*
  - $\mathscr{L}_{T_i} = L_i$
  - $\mathscr{L}_{G_1\,;\,G_2} = \mathscr{L}_{G_1} + \mathscr{L}_{G_2}$
  - $\mathscr{L}_{G_1 \parallel G_2} = \left(\mathscr{L}_{G_1}^{1/\alpha} + \mathscr{L}_{G_2}^{1/\alpha}\right)^{\alpha}$
- *The (unique) optimal schedule* $\mathscr{S}_{\mathrm{PM}}$ *can be computed in polynomial time.*





A tree G (particular SP graph) and the shape of its optimal schedule under any $p(t)$

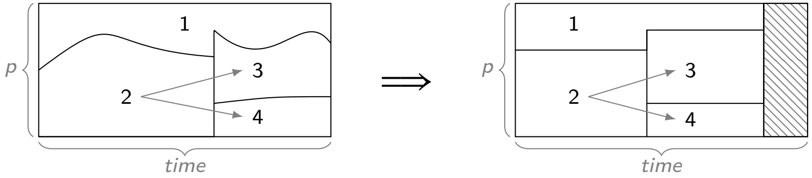
First lemma used in the proof

Definition (Clean interval)

A time interval during which no task terminates in the considered schedule.

Lemma

If $p(t)$ is constant then $p_i(t)$ are constant in optimal schedules.



Modification of a non-optimal schedule

Proof of first lemma

Proof.

Suppose $\mathscr{P}$ optimal where $\exists j,\ p_j(t)$ is not constant on a clean interval Δ

- Let $\mathscr{Q} \approx \mathscr{P}$ except on Δ: $\forall i,\ q_i = \frac{1}{|\Delta|}\int_\Delta p_i(t)dt$ proc. allocated to T_i
- Work done on T_i during Δ:

$$W_i^\Delta(\mathscr{P}) = \int_\Delta p_i(t)^\alpha dt \qquad = |\Delta| \int_{[0,1]} p_i(t_1 + t|\Delta|)^\alpha dt$$

$$W_i^\Delta(\mathscr{Q}) = \int_\Delta \left(\frac{1}{|\Delta|}\int_\Delta p_i(t)dt\right)^\alpha dx = |\Delta| \left(\int_{[0,1]} p_i(t_1 + t|\Delta|)dt\right)^\alpha$$

- Jensen inequality $\Rightarrow \forall i,\ W_i^\Delta(\mathscr{P}) \leq W_i^\Delta(\mathscr{Q})$
  $\Rightarrow \quad W_j^\Delta(\mathscr{P}) < W_j^\Delta(\mathscr{Q})$
- Modify $\mathscr{Q}$: give ε processors to all i from j during $\Delta \longrightarrow \mathscr{Q}$ completes earlier

$\Rightarrow$ $\mathscr{P}$ is not optimal.

□

Proof of first lemma

Proof.

Suppose $\mathscr{P}$ optimal where $\exists j,\ p_j(t)$ is not constant on a clean interval Δ

- Let $\mathscr{Q} \approx \mathscr{P}$ except on Δ: $\forall i,\ q_i = \frac{1}{|\Delta|}\int_\Delta p_i(t)dt$ proc. allocated to T_i
- Work done on T_i during Δ:

$$W_i^\Delta(\mathscr{P}) = \int_\Delta p_i(t)^\alpha dt \qquad = |\Delta| \int_{[0,1]} p_i(t_1 + t|\Delta|)^\alpha dt$$

$$W_i^\Delta(\mathscr{Q}) = \int_\Delta \left(\frac{1}{|\Delta|}\int_\Delta p_i(t)dt\right)^\alpha dx = |\Delta| \left(\int_{[0,1]} p_i(t_1 + t|\Delta|)dt\right)^\alpha$$

- Jensen inequality $\Rightarrow \forall i,\ W_i^\Delta(\mathscr{P}) \leq W_i^\Delta(\mathscr{Q})$
  $\Rightarrow \quad W_j^\Delta(\mathscr{P}) < W_j^\Delta(\mathscr{Q})$
- Modify $\mathscr{Q}$: give ε processors to all i from j during $\Delta \longrightarrow \mathscr{Q}$ completes earlier

$\Rightarrow$ $\mathscr{P}$ is not optimal.

$\square$

Proof of first lemma

Proof.

Suppose $\mathscr{P}$ optimal where $\exists j,\ p_j(t)$ is not constant on a clean interval Δ

- Let $\mathscr{Q} \approx \mathscr{P}$ except on Δ: $\forall i,\ q_i = \frac{1}{|\Delta|}\int_\Delta p_i(t)dt$ proc. allocated to T_i
- Work done on T_i during Δ:

$$W_i^\Delta(\mathscr{P}) = \int_\Delta p_i(t)^\alpha dt \qquad = |\Delta| \int_{[0,1]} p_i(t_1 + t|\Delta|)^\alpha dt$$

$$W_i^\Delta(\mathscr{Q}) = \int_\Delta \left(\frac{1}{|\Delta|}\int_\Delta p_i(t)dt\right)^\alpha dx = |\Delta| \left(\int_{[0,1]} p_i(t_1 + t|\Delta|)dt\right)^\alpha$$

- Jensen inequality $\Rightarrow \forall i,\ W_i^\Delta(\mathscr{P}) \leq W_i^\Delta(\mathscr{Q})$
  $\Rightarrow \quad W_j^\Delta(\mathscr{P}) < W_j^\Delta(\mathscr{Q})$
- Modify $\mathscr{Q}$: give ε processors to all i from j during $\Delta \longrightarrow \mathscr{Q}$ completes earlier

$\Rightarrow$ $\mathscr{P}$ is not optimal.

$\square$

Proof of first lemma

Proof.

Suppose $\mathscr{P}$ optimal where $\exists j,\ p_j(t)$ is not constant on a clean interval Δ

- Let $\mathscr{Q} \approx \mathscr{P}$ except on Δ: $\forall i,\ q_i = \frac{1}{|\Delta|}\int_\Delta p_i(t)dt$ proc. allocated to T_i
- Work done on T_i during Δ:

$$W_i^\Delta(\mathscr{P}) = \int_\Delta p_i(t)^\alpha dt \qquad = |\Delta| \int_{[0,1]} p_i(t_1 + t|\Delta|)^\alpha dt$$

$$W_i^\Delta(\mathscr{Q}) = \int_\Delta \left(\frac{1}{|\Delta|}\int_\Delta p_i(t)dt\right)^\alpha dx = |\Delta| \left(\int_{[0,1]} p_i(t_1 + t|\Delta|)dt\right)^\alpha$$

- Jensen inequality $\Rightarrow \forall i,\ W_i^\Delta(\mathscr{P}) \le W_i^\Delta(\mathscr{Q})$
  $\Rightarrow \quad W_j^\Delta(\mathscr{P}) < W_j^\Delta(\mathscr{Q})$
- Modify $\mathscr{Q}$: give ε processors to all i from j during $\Delta \longrightarrow \mathscr{Q}$ completes earlier

$\Rightarrow$ $\mathscr{P}$ is not optimal.

□

Proof of first lemma

Proof.

Suppose $\mathscr{P}$ optimal where $\exists j,\ p_j(t)$ is not constant on a clean interval Δ

- Let $\mathscr{Q} \approx \mathscr{P}$ except on Δ: $\forall i,\ q_i = \frac{1}{|\Delta|}\int_\Delta p_i(t)dt$ proc. allocated to T_i
- Work done on T_i during Δ:

$$W_i^\Delta(\mathscr{P}) = \int_\Delta p_i(t)^\alpha dt \qquad = |\Delta| \int_{[0,1]} p_i(t_1 + t|\Delta|)^\alpha dt$$

$$W_i^\Delta(\mathscr{Q}) = \int_\Delta \left(\frac{1}{|\Delta|}\int_\Delta p_i(t)dt\right)^\alpha dx = |\Delta| \left(\int_{[0,1]} p_i(t_1 + t|\Delta|)dt\right)^\alpha$$

- Jensen inequality $\Rightarrow \forall i,\ W_i^\Delta(\mathscr{P}) \leq W_i^\Delta(\mathscr{Q})$
  $\Rightarrow \quad W_j^\Delta(\mathscr{P}) < W_j^\Delta(\mathscr{Q})$
- Modify $\mathscr{Q}$: give ε processors to all i from j during $\Delta \longrightarrow \mathscr{Q}$ completes earlier

$\Rightarrow$ $\mathscr{P}$ is not optimal.

□

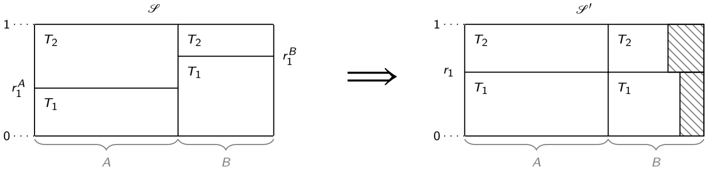
Main lemma

Lemma

In optimal schedules of $G = T_1 \parallel T_2$, $p_1(t)/p(t)$ is constant.

Proof scheme. *(Note that $p(t)$ is not necessarily constant)*

- Suppose that $\mathscr{S}$ optimal and $p_1(t)/p(t)$ is not constant
- We can transform $\mathscr{S}$ in $\mathscr{S}'$ with a smaller makespan
- Properties used: strict concavity of f and $\forall xy,\ f(xy) = f(x)f(y)$



$\square$

End of the proof of the theorem

Few steps remaining to prove the theorem

- $T_1 \parallel T_2$ under any $p(t)$ $\iff$ $T_{1\parallel 2}$ of length $\mathscr{L}_{1\parallel 2}$ under any $p(t)$
- $T_1 \,;\, T_2$ under any $p(t)$ $\iff$ $T_{1;2}$ of length $\mathscr{L}_{1;2}$ under any $p(t)$
- Proof by induction on the structure of G

End of the proof of the theorem

Few steps remaining to prove the theorem

- $T_1 \parallel T_2$ under any $p(t)$ $\iff$ $T_{1\parallel 2}$ of length $\mathcal{L}_{1\parallel 2}$ under any $p(t)$
- $T_1 ; T_2$ under any $p(t)$ $\iff$ $T_{1;2}$ of length $\mathcal{L}_{1;2}$ under any $p(t)$
- Proof by induction on the structure of G

Computing x^α and $x^{1/\alpha}$

Claim: The optimal solution can be computed in polynomial-time

Issue: Need to compute functions $x \mapsto x^\alpha$ and $x \mapsto x^{1/\alpha}$

Recall: $\mathcal{L}_{G_1 \parallel G_2} = \left(\mathcal{L}_{G_1}^{1/\alpha} + \mathcal{L}_{G_2}^{1/\alpha}\right)^\alpha$

Hypothesis: Considered as elementary operations
(or similarly consider polynomial-time approximations)

Outline

1 Introduction and notations

2 Minimizing the makespan

3 Extensions to distributed memory
- Homogeneous multicore nodes
- Heterogeneous multicore nodes

4 Gain of speedup-aware strategies

5 Conclusion

Description of the distributed memory model

Motivation

- Multiprocessors nodes
- Each node has its own memory
- $\mathscr{R}$ constraint: tasks cannot be split between two nodes

Two special cases

- Processing power:
  - two identical nodes of size p
  - two nodes of sizes p and q
- *Same hypothesis on the computation of x^α and $x^{1/\alpha}$*

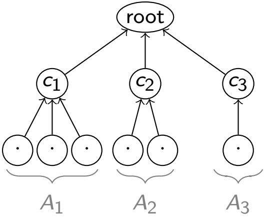
Results

- Makespan-minimizing schedules:
  - NP-complete for independent tasks and identical nodes
  - Identical nodes: $\left(\frac{4}{3}\right)^\alpha$-approximation for any tree
  - Different nodes: FPTAS for independent tasks
- Strategy: adaptations from the PM schedule (previous section) without $\mathscr{R}$

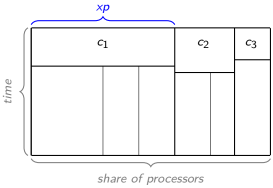
The identical nodes problem

Notations of the tree G

- Root has length 0 and several children
- Children of the root: c_i
- A_i: Subtree rooted at c_i
- $\mathscr{L}_{A_1} \geq \mathscr{L}_{A_2} \geq \cdots \geq \mathscr{L}_{A_k}$
- A_1 is allocated xp processors by the PM schedule launched on $2p$ processors



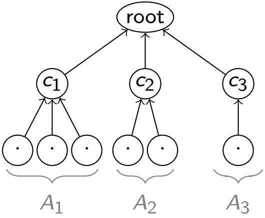
PM schedule on $2p$ processors



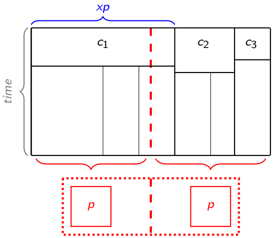
The identical nodes problem

Notations of the tree G

- Root has length 0 and several children
- Children of the root: c_i
- A_i: Subtree rooted at c_i
- $\mathcal{L}_{A_1} \geq \mathcal{L}_{A_2} \geq \cdots \geq \mathcal{L}_{A_k}$
- A_1 is allocated xp processors by the PM schedule launched on $2p$ processors



PM schedule on $2p$ processors

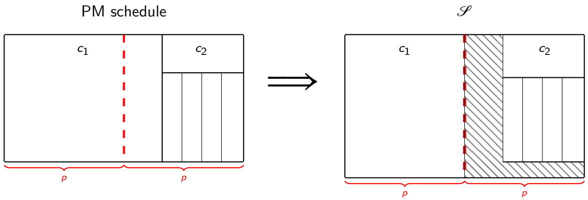


The $\left(\frac{4}{3}\right)^{\alpha}$-approximation algorithm

```
function HomogeneousApp(G, p)
    Compute the PM schedule 𝒮_PM of G on 2p processors
    if x ≥ 1 and c_1 is a leaf then
        Build 𝒮 from 𝒮_PM: shrink c_1 → p processors
    else if x ≤ 1 then                              ▷ Case implying the (4/3)^α factor
        Build 𝒮: partition the A_i's in both nodes
    else
        Compute the schedule 𝒮_p
        𝒮^r ← HomogeneousApp((A_1 \ c_1) ∥ B̄_p , p)
        Build 𝒮: schedule (A_1 \ c_1) ∥ B̄_p as in 𝒮^r then c_1 ∥ B_p as in 𝒮_p
    return 𝒮
```

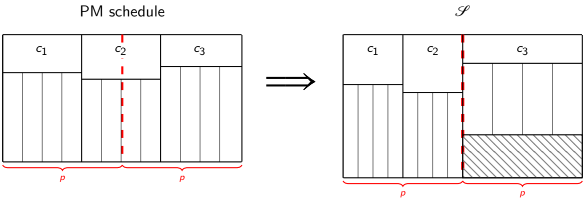
The $\left(\frac{4}{3}\right)^{\alpha}$-approximation algorithm

function $\textsc{HomogeneousApp}(G, p)$
    Compute the PM schedule $\mathscr{S}_{\mathrm{PM}}$ of G on $2p$ processors
    if $x \geq 1$ and c_1 is a leaf **then**
        ▶ Build $\mathscr{S}$ from $\mathscr{S}_{\mathrm{PM}}$: shrink $c_1 \to p$ processors
    else if $x \leq 1$ **then** ▷ Case implying the $\left(\frac{4}{3}\right)^{\alpha}$ factor
        Build $\mathscr{S}$: partition the A_i's in both nodes
    else
        Compute the schedule $\mathscr{S}_p$
        $\mathscr{S}^r \leftarrow \textsc{HomogeneousApp}((A_1 \setminus c_1) \parallel \overline{B}_p\ ,\ p)$
        Build $\mathscr{S}$: schedule $(A_1 \setminus c_1) \parallel \overline{B}_p$ as in $\mathscr{S}^r$ then $c_1 \parallel B_p$ as in $\mathscr{S}_p$
    return $\mathscr{S}$



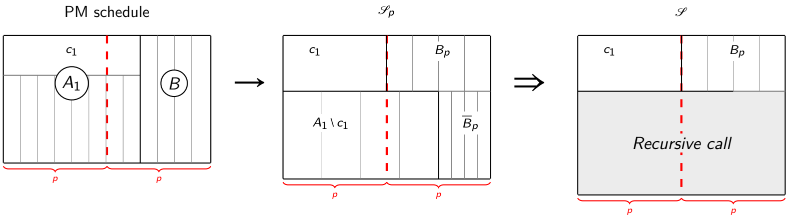
The $\left(\frac{4}{3}\right)^{\alpha}$-approximation algorithm

function HomogeneousApp(G, p)
    Compute the PM schedule $\mathscr{S}_{\mathrm{PM}}$ of G on $2p$ processors
    if $x \geq 1$ and c_1 is a leaf **then**
        Build $\mathscr{S}$ from $\mathscr{S}_{\mathrm{PM}}$: shrink $c_1 \rightarrow p$ processors
    else if $x \leq 1$ **then** ▷ Case implying the $\left(\frac{4}{3}\right)^{\alpha}$ factor
        ▶ Build $\mathscr{S}$: partition the A_i's in both nodes
    else
        Compute the schedule $\mathscr{S}_p$
        $\mathscr{S}^r \leftarrow$ HomogeneousApp$((A_1 \setminus c_1) \parallel \overline{B}_p\ ,\ p)$
        Build $\mathscr{S}$: schedule $(A_1 \setminus c_1) \parallel \overline{B}_p$ as in $\mathscr{S}^r$ then $c_1 \parallel B_p$ as in $\mathscr{S}_p$
    return $\mathscr{S}$



The $\left(\frac{4}{3}\right)^{\alpha}$-approximation algorithm

function $\textsc{HomogeneousApp}(G, p)$
  Compute the PM schedule $\mathscr{S}_{\mathrm{PM}}$ of G on $2p$ processors
  if $x \geq 1$ and c_1 is a leaf **then**
    Build $\mathscr{S}$ from $\mathscr{S}_{\mathrm{PM}}$: shrink $c_1 \rightarrow p$ processors
  else if $x \leq 1$ **then** ▷ Case implying the $\left(\frac{4}{3}\right)^{\alpha}$ factor
    Build $\mathscr{S}$: partition the A_i's in both nodes
  else
    Compute the schedule $\mathscr{S}_p$
    $\mathscr{S}^r \leftarrow \textsc{HomogeneousApp}((A_1 \setminus c_1) \parallel \overline{B}_p\ ,\ p)$
    ▶ Build $\mathscr{S}$: schedule $(A_1 \setminus c_1) \parallel \overline{B}_p$ as in $\mathscr{S}^r$ then $c_1 \parallel B_p$ as in $\mathscr{S}_p$
  return $\mathscr{S}$



Heterogeneous multicore nodes

Definition of (p,q)-SCHEDULING

- Two nodes of size p and q with the same α value
- n independent tasks $T_1 \dots T_n$ of length L_i
- Objective: map each task to a node to minimize the makespan
- Approximation: given ε, find a schedule whose makespan is $\leq (1+\varepsilon)\mathrm{OPT}$

Related problem: SUBSET SUM

- `Input` : n integers and a target s
  `Output`: a subset of the integers that sums to the largest number $\leq s$
- FPTAS [Kellerer03]: given ε, returns a solution whose sum is $\geq (1-\varepsilon)\mathrm{OPT}$

Theorem

There exists a FPTAS to (p,q)-SCHEDULING RESTRICTED where $L_i^{1/\alpha}$ are integer.

Outline

1. Introduction and notations
2. Minimizing the makespan
3. Extensions to distributed memory
4. Gain of speedup-aware strategies
5. Conclusion

Evaluation method

Strategies to compare to PM

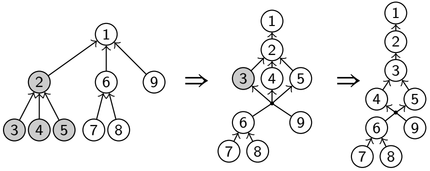
- DIVISIBLE: sequential schedule
- PROPORTIONAL (proportional mapping): power allocated to each subtree is proportional to its length (eq. to Musicus assuming $\alpha = 1$)

Dataset

- 600 trees containing between 2,000 and 1,000,000 nodes
- $p(t) = 40$ or $p(t) = 100$, $\alpha \in [0.5, 1]$
- Speedup: p^{α} for $p \geq 1$ and p otherwise

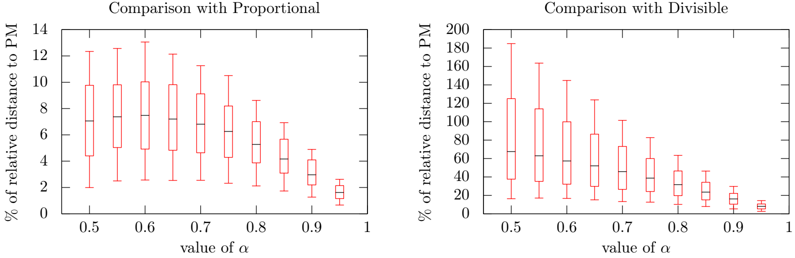
Obtention of the dataset:

- Compute assembly trees for a set of the University of Florida Sparse Matrix Collection
- Modify the trees such that PM allocates ≥ 1 processors per task



Shaded tasks are alloted < 1 processor by PM

Results



Comparison to the PM schedule with $p(t) = 40$

Expected gain of 3%–5% for $\alpha \in [0.85, 0.95]$ compared to PROPORTIONAL

- Still noticeable gain if transposed to real software implementations
- α should be smaller for machines with smaller memory bandwidth
- Core computing rates increase faster than memory bandwidth
  → lower values of α are expected

Outline

1. Introduction and notations
2. Minimizing the makespan
3. Extensions to distributed memory
4. Gain of speedup-aware strategies
5. Conclusion

Conclusion

Scheduling malleable task trees on multicore platforms with speedup $= p^\alpha$

- Model motivated and validated by experiments: $\alpha \in [0.85, 0.95]$
- Intuitive proof of the optimal scheduling strategy

Extension to two multicore nodes

- NP-completeness of the problem
- $\left(\frac{4}{3}\right)^\alpha$-approximation for trees on homogeneous nodes
- FPTAS for independent tasks on heterogeneous nodes

Perspectives

- Handle several heterogeneous nodes
- Handle nodes with different values of α (accelerators: GPU, Xeon Phi)
- Implement the PM allocation scheme in a sparse solver