The 2011 International Conference on
High Performance Computing & Simulation

Workshop on Optimization Issues in Energy Efficient Distributed Systems

Improving Power efficiency of Dense Linear Algebra Algorithms on Multi-Core Processors via Slack Control

Pedro Alonso[1], Manuel F. Dolz[2], Rafael Mayo[2], Enrique S. Quintana-Ortí[2]

1 Universitat Politècnica de València

2 Universitat Jaume I

July 4–8, 2011, Istanbul (Turkey)

Motivation

- High performance computing:
  - Optimization of algorithms applied to solve complex problems
- Technological advance ⇒ improve performance:
  - Processors works at higher frequencies
  - Higher number of cores per socket (processor)
- Large number of processors and cores ⇒ High energy consumption
- Methods, algorithms and techniques to reduce energy consumption applied to high performance computing.
  - Reduce the frequency of processors with DVFS technique

Outline

Introduction

- **Scheduling tasks** of dense linear algebra algorithms
  - Examples: Cholesky, QR and LU factorizations
- **Energy saving tools** available for multi-core processors
  - Example: Dynamic Voltage and Frequency Scaling (DVFS)

Scheduling tasks + DVFS

$\Downarrow$

Power-aware scheduling on multi-core processors

- **Our strategy**: Reduce the frequency of cores that will execute non-critical tasks to decrease idle times without sacrifying total performance of the algorithm

$\Downarrow$

Energy saving

Introduction

- **Scheduling tasks** of dense linear algebra algorithms
  - Examples: Cholesky, QR and LU factorizations
- **Energy saving tools** available for multi-core processors
  - Example: Dynamic Voltage and Frequency Scaling (DVFS)

Scheduling tasks + DVFS

$\Downarrow$

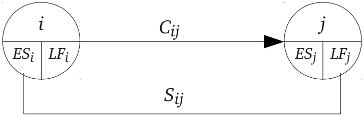
Power-aware scheduling on multi-core processors

- **Our strategy**: Reduce the frequency of cores that will execute non-critical tasks to decrease idle times without sacrifying total performance of the algorithm

$\Downarrow$

Energy saving

The Critical Path Method



$ES_i = max(ES_k + C_{ki})$ $\quad$ $LF_j = min(LF_k + C_{jk})$

$S_{ij} = ES_j - ES_i - C_{ij}$

Concepts:

- **DAG of dependencies**
  - Nodes $\Rightarrow$ Temporal events
  - Edges $\Rightarrow$ Tasks
- **Times**
  - Early and latest times to start and finalize execution of tasks
- **Total slack**:
  - Amount of time that a task can be delayed without increasing the total execution time of the algorithm
- **Critical path:**
  - Formed by a succession of tasks, from initial to final node of the graph, with total slack = 0.

Application to dense linear algebra algorithms

Objective $\Rightarrow$ obtain the dependency graph corresponding to the computation of a dense linear algebra algorithm, apply the Critical Path Method to analize slacks and reducing them with our Slack Reduction Algorithm

Example: Cholesky factorization of a matrix consisting of 3×3 blocks

for $k = 1, 2, \ldots, s$ **do**		
$A_{kk} = L_{kk} L_{kk}^T$	CHOLESKY FACTORIZATION	$\frac{b^3}{3}$ flops $\rightsquigarrow$ 0,33 $u.t.$
for $i = k+1, k+2, \ldots, s$ **do**		
$A_{ik} \leftarrow A_{ik} L_{kk}^{-T}$	TRIANGULAR SYSTEM SOLVE	b^3 flops $\rightsquigarrow$ 1 $u.t.$
end for		
for $i = k+1, k+2, \ldots, s$ **do**		
for $j = k+1, k+2, \ldots, i-1$ **do**		
$A_{ij} \leftarrow A_{ij} - A_{ik} A_{jk}^T$	MATRIX-MATRIX PRODUCT	$2b^3$ flops $\rightsquigarrow$ 2 $u.t.$
end for		
$A_{ii} \leftarrow A_{ii} - A_{ik} A_{ik}^T$	SYMMETRIC RANK-b UPDATE	b^3 flops $\rightsquigarrow$ 1 $u.t.$
end for		
end for		

Application to dense linear algebra algorithms

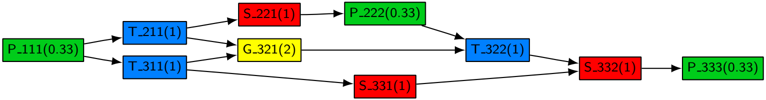
Objective $\Rightarrow$ obtain the dependency graph corresponding to the computation of a dense linear algebra algorithm, apply the Critical Path Method to analize slacks and reducing them with our Slack Reduction Algorithm

Example: Cholesky factorization of a matrix consisting of 3×3 blocks

for $k = 1, 2, \ldots, s$ **do**
  $A_{kk} = L_{kk} L_{kk}^T$ — CHOLESKY FACTORIZATION — $\frac{b^3}{3}$ flops $\rightsquigarrow$ 0,33 $u.t.$
  for $i = k+1, k+2, \ldots, s$ **do**
    $A_{ik} \leftarrow A_{ik} L_{kk}^{-T}$ — TRIANGULAR SYSTEM SOLVE — b^3 flops $\rightsquigarrow$ 1 $u.t.$
  end for
  for $i = k+1, k+2, \ldots, s$ **do**
    for $j = k+1, k+2, \ldots, i-1$ **do**
      $A_{ij} \leftarrow A_{ij} - A_{ik} A_{jk}^T$ — MATRIX-MATRIX PRODUCT — $2b^3$ flops $\rightsquigarrow$ 2 $u.t.$
    end for
    $A_{ii} \leftarrow A_{ii} - A_{ik} A_{ik}^T$ — SYMMETRIC RANK-b UPDATE — b^3 flops $\rightsquigarrow$ 1 $u.t.$
  end for
end for

Application to dense linear algebra algorithms

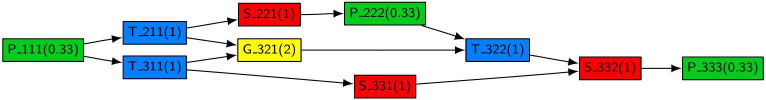
- Taks-node DAG capturing the data dependencies in the computation of the Cholesky factorization of a matrix consisting of 3 × 3 blocks



- Graph transformation in order to apply CPM
- Conversion from **task-node** to **task-edge** graph

Application to dense linear algebra algorithms

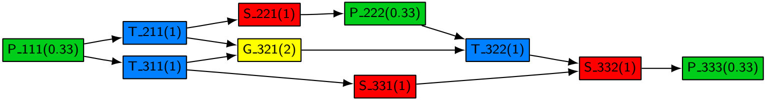
- Taks-node DAG capturing the data dependencies in the computation of the Cholesky factorization of a matrix consisting of 3×3 blocks



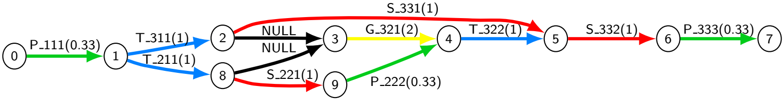
- Graph transformation in order to apply CPM
- Conversion from **task-node** to **task-edge** graph

Application to dense linear algebra algorithms

- Taks-node DAG capturing the data dependencies in the computation of the Cholesky factorization of a matrix consisting of 3×3 blocks



- Graph transformation in order to apply CPM
- Conversion from **task-node** to **task-edge** graph

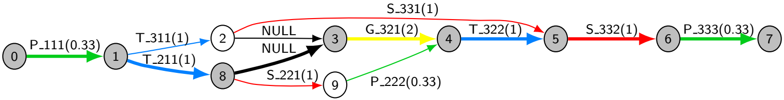


Application to dense linear algebra algorithms

Application of CPM to the task-edge DAG of the Cholesky factorization of a matrix consisting of 3×3 blocks

Task	$i - j$	$C_{i,j}$	ES_i	LF_j	$S_{i,j}$
P_111	0-1	0.33	0	0.33	0
T_211	1-8	1	0.33	1.33	0
T_311	1-2	1	0.33	1.33	0
NULL	2-3	0	1.33	1.33	0
S_221	8-9	1	1.33	3	0.67
G_321	3-4	2	1.33	3.33	0
S_331	2-5	1	1.33	4.33	2
P_222	9-4	0.33	2.33	3.33	0.67
T_322	4-5	1	3.33	4.33	0
S_332	5-6	1	4.33	5.33	0
P_333	6-7	0.33	5.33	5.67	0
NULL	8-3	0	1.33	1.33	0

Critical path:



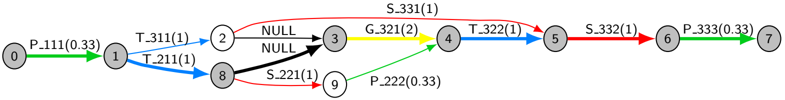
Objective: tune the slack of those tasks with $S_{i,j} > 0$, reducing its execution frequency and yielding low power usage → *Slack Reduction Algorithm*

Application to dense linear algebra algorithms

Application of CPM to the task-edge DAG of the Cholesky factorization of a matrix consisting of 3×3 blocks

Task	$i - j$	$C_{i,j}$	ES_i	LF_j	$S_{i,j}$
P_111	0-1	0.33	0	0.33	0
T_211	1-8	1	0.33	1.33	0
T_311	1-2	1	0.33	1.33	0
NULL	2-3	0	1.33	1.33	0
S_221	8-9	1	1.33	3	0.67
G_321	3-4	2	1.33	3.33	0
S_331	2-5	1	1.33	4.33	2
P_222	9-4	0.33	2.33	3.33	0.67
T_322	4-5	1	3.33	4.33	0
S_332	5-6	1	4.33	5.33	0
P_333	6-7	0.33	5.33	5.67	0
NULL	8-3	0	1.33	1.33	0

Critical path:



Objective: tune the slack of those tasks with $S_{i,j} > 0$, reducing its execution frequency and yielding low power usage → *Slack Reduction Algorithm*

Slack reduction

Slack reduction algorithm

1. Frequency assignment
2. Critical subpath extraction
3. Slack reduction

1 Frequency assignment

Example: Cholesky factorization of 3×3 blocks:

- Discrete collection of frequencies: {2.27, 2.13, 2.00, 1.87, 1.73, 1.60} GHz
- The execution time of tasks increase inversely proportional as its frequency decreases!

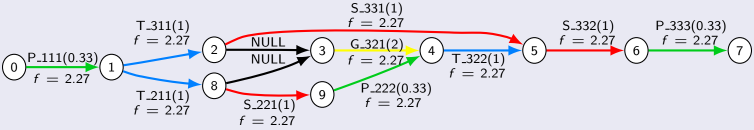
Slack reduction

Slack reduction algorithm

1. Frequency assignment
2. Critical subpath extraction
3. Slack reduction

1 Frequency assignment

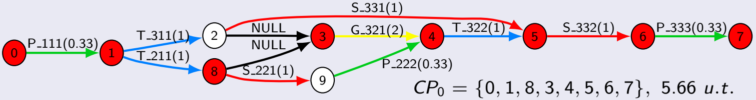
Example: Cholesky factorization of 3×3 blocks:



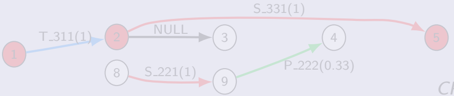
- Discrete collection of frequencies: {2.27, 2.13, 2.00, 1.87, 1.73, 1.60} GHz
- The execution time of tasks increase inversely proportional as its frequency decreases!

Critical subpath extraction

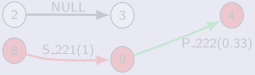
2 Critical subpath extraction



$CP_0 = \{0, 1, 8, 3, 4, 5, 6, 7\},\ 5.66\ u.t.$



$CP_1 = \{1, 2, 5\},\ 2\ u.t.$

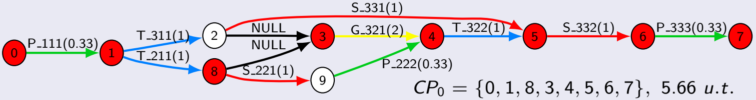


$CP_2 = \{8, 9, 4\},\ 1.33\ u.t.$

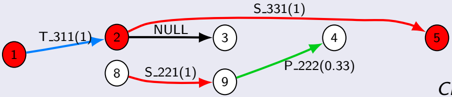
$CP_3 = \{2, 3\},\ 0\ u.t.$

Critical subpath extraction

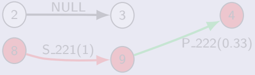
2 Critical subpath extraction



$CP_0 = \{0, 1, 8, 3, 4, 5, 6, 7\},\ 5.66\ u.t.$



$CP_1 = \{1, 2, 5\},\ 2\ u.t.$

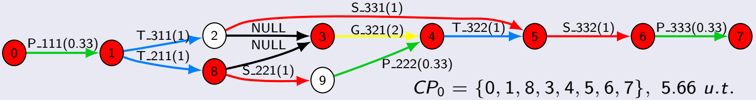


$CP_2 = \{8, 9, 4\},\ 1.33\ u.t.$

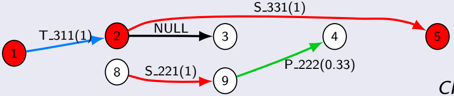
$CP_3 = \{2, 3\},\ 0\ u.t.$

Critical subpath extraction

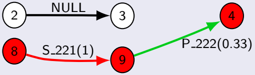
2 Critical subpath extraction



$$CP_0 = \{0, 1, 8, 3, 4, 5, 6, 7\}, \ 5.66 \ u.t.$$



$$CP_1 = \{1, 2, 5\}, \ 2 \ u.t.$$

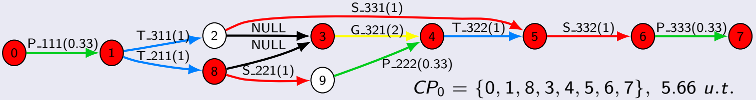


$$CP_2 = \{8, 9, 4\}, \ 1.33 \ u.t.$$

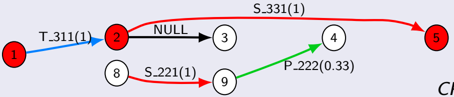
$$CP_3 = \{2, 3\}, \ 0 \ u.t.$$

Critical subpath extraction

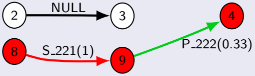
2 Critical subpath extraction



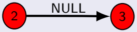
$CP_0 = \{0, 1, 8, 3, 4, 5, 6, 7\}, \; 5.66 \; u.t.$



$CP_1 = \{1, 2, 5\}, \; 2 \; u.t.$



$CP_2 = \{8, 9, 4\}, \; 1.33 \; u.t.$



$CP_3 = \{2, 3\}, \; 0 \; u.t.$

Slack Reduction Algorithm (I)

Iteration 1

Process critical subpath $CP_1 = \{1, 2, 5\}$:

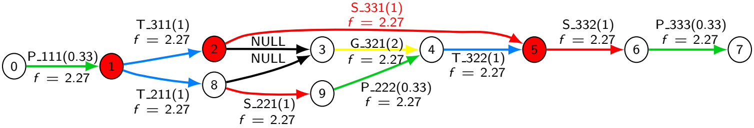
1. Checks for tasks of CP_1 with a nonzero slack: only task S_331
2. Slack is reduced by reducing execution frequency of task:
   - S_331: 2.27 GHz ⇒ 1.60 GHz; 1 u.t. ⇒ 1.42 u.t.; Slack 2 u.t.⇒ 0.58 u.t.

Slack Reduction Algorithm (I)

Iteration 1

Process critical subpath $CP_1 = \{1, 2, 5\}$:

1. Checks for tasks of CP_1 with a nonzero slack: only task S_331
2. Slack is reduced by reducing execution frequency of task:
   - S_331: 2.27 GHz ⇒ 1.60 GHz; 1 u.t. ⇒ 1.42 u.t.; Slack 2 u.t.⇒ 0.58 u.t.

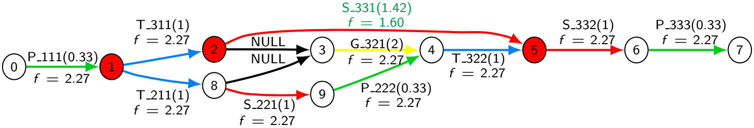


Slack Reduction Algorithm (II)

Iteration 1

Process critical subpath $CP_1 = \{1, 2, 5\}$:

1. Checks for tasks of CP_1 with a nonzero slack: only task S_331
2. Slack is reduced by reducing execution frequency of task:
   - S_331: 2.27 GHz ⇒ 1.60 GHz; 1 u.t. ⇒ 1.42 u.t.; Slack 2 u.t.⇒ 0.58 u.t.



Slack Reduction Algorithm (III)

Iteration 2

Process of critical subpath $CP_2 = \{8, 9, 4\}$:

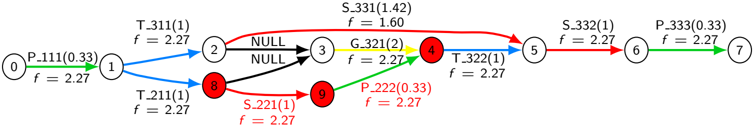
1. Checks for tasks of CP_2 with a nonzero slack: tasks S_221 and P_222
2. Slack is equaly splitted in both tasks:
   - S_221: 2.27 GHz ⇒ 1.73 GHz; 1 u.t. ⇒ 1.31 u.t.; Slack 0.67 u.t.⇒ 0 u.t.
   - P_222: 2.27 GHz ⇒ 1.73 GHz; 0.33 u.t. ⇒ 0.44 u.t.; Slack 0.67 u.t.⇒ 0 u.t.

Slack Reduction Algorithm (III)

Iteration 2

Process of critical subpath $CP_2 = \{8, 9, 4\}$:

1. Checks for tasks of CP_2 with a nonzero slack: tasks S_221 and P_222
2. Slack is equaly splitted in both tasks:
   - S_221: 2.27 GHz ⇒ 1.73 GHz; 1 u.t. ⇒ 1.31 u.t.; Slack 0.67 u.t.⇒ 0 u.t.
   - P_222: 2.27 GHz ⇒ 1.73 GHz; 0.33 u.t. ⇒ 0.44 u.t.; Slack 0.67 u.t.⇒ 0 u.t.

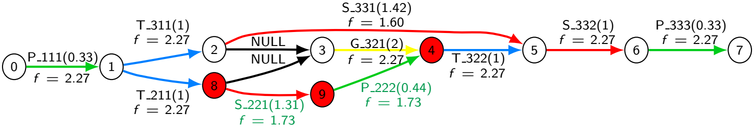


Slack Reduction Algorithm (IV)

Iteration 2

Process of critical subpath $CP_2 = \{8, 9, 4\}$:

1. Checks for tasks of CP_2 with a nonzero slack: tasks S_221 and P_222
2. Slack is equaly splitted in both tasks:
   - S_221: 2.27 GHz ⇒ 1.73 GHz; 1 u.t. ⇒ 1.31 u.t.; Slack 0.67 u.t.⇒ 0 u.t.
   - P_222: 2.27 GHz ⇒ 1.73 GHz; 0.33 u.t. ⇒ 0.44 u.t.; Slack 0.67 u.t.⇒ 0 u.t.



Slack Reduction Algorithm (V)

Iteration 3

Process of critical subpath $CP_2 = \{2, 3\}$:

1. Requires no processing: subpath only contains a NULL task

Frequency assignment and cost of the task-edge DAG of dependencies of Cholesky algorithm consisting of a matrix 3×3:

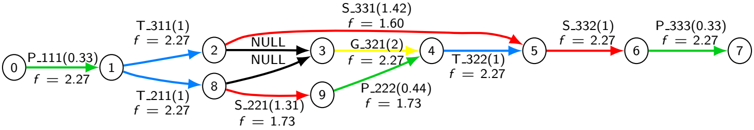
Slack Reduction Algorithm (V)

Iteration 3

Process of critical subpath $CP_2 = \{2, 3\}$:

1. Requires no processing: subpath only contains a `NULL` task

Frequency assignment and cost of the task-edge DAG of dependencies of Cholesky algorithm consisting of a matrix 3×3:



Simulator (I)

Simulator to evaluate the performance of our strategy

Input parameters:

- DAG capturing tasks and dependencies of a blocked algorithm and frequencies recommended by Slack Reduction Algorithm
- A simple description of the target architecture:
  - Number of sockets (physical processors)
  - Number of cores per socket
- Discrete range of frequencies and its associated voltages
- The cost (overhead) required to perform frequency changes

Static priority list scheduler:

- Duration of tasks is known in advance
- Tasks that lie on critical path must be prioritized

Benchmark algorithms

- Blocked algorithms: Cholesky and QR with incremental pivoting
  - Block size: $b = 192$
  - Matrix size varies from 576 to 2,112
- Target architecture
  - Four quad-core sockets (a total of 16 cores)
  - Discrete range of frequencies: $\{2.27, 2.13, 2.00, 1.87, 1.73, 1.60\}$ GHz
  - Associated voltages vary from 0.75 to 1.35 V (linear relation between voltage and the frequency)
  - Frequency change latency: 0.1 u.t.
  - Representative values from Intel Xeon 5520 processor
- Metrics:

Execution time (u.t.)

- $T_{SRAPolicy}$
- $T_{Nopolicy}$
- Impact of SRA on time

$$\boxed{\%T_{SRA} = \frac{T_{SRAPolicy}}{T_{Nopolicy}} \cdot 100}$$

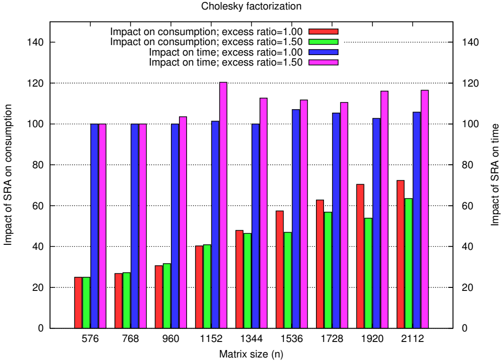
Consumption (u.c.)

- $C_{SRAPolicy} = \sum_{i=1}^{n} v_i^2 \; T(f_i) + v^2 * \; T(f_{max})$
- $C_{NoPolicy} = v^2 \; T(f_{max})$
- Impact of SRA on consumption

$$\boxed{\%C_{SRA} = \frac{C_{SRAPolicy}}{C_{NoPolicy}} \cdot 100}$$

Cholesky factorization

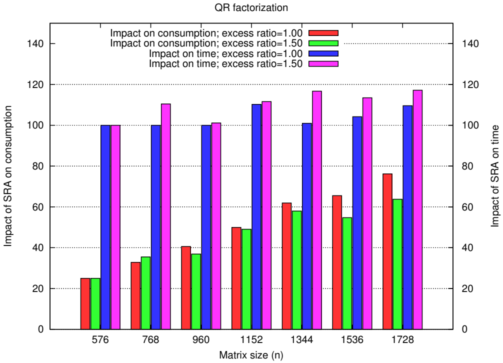
Impact of the SRA on energy and time for the Cholesky factorization:



- Excess ratio (e=1): Time is not compromised in some cases and increases consumption with matrix size
- Excess ratio (e=1.5): Time is compromised in most cases but there is less consumption than with $e = 1$

QR Factorization

Impact of the SRA on energy and time for the QR factorization:



- Excess ratio=1: Time is not compromised in some cases but consumption increases with matrix size
- Excess ratio=1.5: Time is compromised in most cases but there is less consumption than Excess ratio=1

Conclusions

- Idea: exploit task-level parallelism to reduce energy consumption
- Objective: to reduce idle times by reducing execution frequency of tasks
  - Slack Reduction Algorithm
    - Tasks with slack are executed at a minor frequency
  - Theoretical results of dense linear algorithms
    - Cholesky and QR (with incremental pivoting) factorizations
    - Significant reduction in power consumption under realistic conditions
    - A higher ratio between number of computational resources and number of tasks yields a more reduced power consumption
    - LU factorization show similar behaviour to that of QR factorization.

Future work

- Some improvements:
  - Slack Reduction Algorithm is a static strategy but... it has an implicit cost!
  - We are working in dynamic strategies to work at run-time for adapt frequency and reduce slacks
- Future work:
  - We plan to integrate these theoretical study into a real run-time scheduler, e.g., SuperMatrix as part of `libflame`

Thanks for your attention!

Questions?