Analysis of Dynamic Scheduling Strategies for Matrix Multiplication on Heterogeneous Platforms

Olivier Beaumont and Loris Marchal
(INRIA, Bordeaux) (CNRS, Lyon)

NCST workshop, April 2nd, 2014.
Outline

Introduction

Outer Product
  Static strategies to minimize communications
  Randomized Dynamic Strategies

Matrix Product

Conclusion and Perspectives
Outline

Introduction

Outer Product
  Static strategies to minimize communications
  Randomized Dynamic Strategies

Matrix Product

Conclusion and Perspectives
Scheduling and resource allocation problems are hard due to
- Failures
- Non-determinism of execution/transfer time
- Heterogeneity

All this makes static strategies useless

Deciding in advance:
- Where to place tasks
- When to execute them

Can make the system very slow
Solutions

- On the theoretical side: online scheduling, robust scheduling
  - Given probability distribution of execution/transfer time
  - Find the allocation/schedule that minimizes expected makespan
  - Worst case is also interesting and not trivial
  - Bad point: extremely difficult

- On the practical side:
  - Mostly dynamic, demand driven strategies (e.g. Hadoop, PaRSEC, StarSs, KAALP, StarPU)
  - Based on rough estimation of execution time
  - Still, open questions:
    - Does it work so well (simulation part)?
    - Why does it work (theoretical part)?
On the application side

- More and more applications expressed as independent tasks:
  - MapReduce, Divisible Load
  - Independent tasks are what dynamic schedulers see

- Lessons:
  - Placing tasks close to the actual data location is crucial
  - Placing tasks on well adapted resources is crucial
  - Deciding task order is not crucial, provided that there is no idle time.
What is our goal?

- Propose analytical models for dynamic strategies

- Analyze dynamic strategies to understand:
  - What makes dynamic strategies efficient?
  - What can be done to improve them?

- For basic applications first
  - Independent tasks, MapReduce
  - Linear algebra without dependencies (except data dep.)

- Today: outer product (and matrix multiplication)
Outline

Introduction

Outer Product
  Static strategies to minimize communications
  Randomized Dynamic Strategies

Matrix Product

Conclusion and Perspectives
Outer Product

- Basic $O(N^2)$ operation
  - 2 vectors of size $N$: $a$ and $b$
  - Output $M = ab^T$, i.e. all $M_{i,j} = a_i \cdot b_j$ values, $1 \leq i, j \leq N$
- If $P_k$ is responsible for computing the red $M_{i,j}$ values it needs to store the red values of $a$ and $b$

![Diagram showing the computation of outer product and its communication requirements.](image)
Outer Product : Communication Needs

- Load-balancing ensured by demand-driven strategy
- Overlap communication/computation
- Need to replicate data
- Focus on minimizing communication amount
- Consider first homogeneous processors

Let us denote by $V$ the total volume of data sent to all processors

- Data sent to processor $P_k$: $\frac{V}{P}$
- Workload at processor $P_k$: $\left(\frac{V}{2P}\right)^2$
- Total work is $N^2$
- So that $P \left(\frac{V}{2P}\right)^2 = N^2$ and $V = 2N\sqrt{P}$
- $\sqrt{P}$ is the smallest achievable replication ratio
The $\sqrt{P}$ lower bound does not hold in the heterogeneous case.

Let $s_k$ denote the speed of $P_k$.

- Load balancing, work for $P_k$:
  \[ w_k = x_k \times y_k = \frac{s_k}{\sum_i s_i} N^2 \]
- Amount of data: \[ V_k = x_k + y_k \]

Problem: partition the square $[1; N] \times [1; N]$ so that

- \[ x_k \times y_k = \frac{s_k}{\sum_i s_i} N^2 \]
- \[ \sum_k x_k + y_k \text{ is minimized} \]

Lower bound:
\[ 2N \sum_k \sqrt{\frac{s_k}{\sum_i s_i}} \]

Well known combinatorial problem
- 7/4-approximation algorithm
Previous approximation algorithm:
  - Very static
  - Not well suited to large-scale dynamic faulty systems
  - Processor speeds are hard to predict

More dynamic strategies are needed
  - Demand-driven strategies
  - Processors request blocks when needed (or slightly before)
  - We assume there is a central scheduler/dispatcher
  - Question: which task to send to which processor?
Randomized Dynamic Strategies

Basic strategies:

- When requested for a new task, send a random one \((\text{RandomOuter})\)
- Send tasks by rows \((\text{SortedOuter})\)

Expect to lead to large amount of communications, because of data replication

Simple data-aware strategy \(\text{DynamicOuter}\):

- When a processor \(P_k\) request a task, send new \(a_i\) and \(b_j\) to \(P_k\)
- Allocate all unprocessed tasks \(a_i \times b_{j'}\) (for \(b_{j'}\) already on \(P_k\))
- Allocate all unprocessed tasks \(a_{i'} \times b_j\) (for \(a_{i'}\) already on \(P_k\))

(small scheduling overhead: maintain sets of \(a\) and \(b\) on each processors)
Randomized Dynamic Strategies

Simple data-aware strategy \textsc{DynamicOuter}:

- When a processor $P_k$ requests a task, send new $a_i$ and $b_j$ to $P_k$
- Allocate all unprocessed tasks $a_i \times b_{j'}$ (for $b_{j'}$ already on $P_k$)
- Allocate all unprocessed tasks $a_{i'} \times b_j$ (for $a_{i'}$ already on $P_k$)

(small scheduling overhead: maintain sets of $a$ and $b$ on each processors)
Randomized Dynamic Strategies

Simple data-aware strategy **DynamicOuter**:

- When a processor $P_k$ requests a task, send new $a_i$ and $b_j$ to $P_k$
- Allocate all unprocessed tasks $a_i \times b_{j'}$ (for $b_{j'}$ already on $P_k$)
- Allocate all unprocessed tasks $a_{i'} \times b_j$ (for $a_{i'}$ already on $P_k$)

(small scheduling overhead: maintain sets of $a$ and $b$ on each processors)
Randomized Dynamic Strategies

Simple data-aware strategy **DynamicOuter**:  
- When a processor $P_k$ requests a task, send new $a_i$ and $b_j$ to $P_k$  
- Allocate all unprocessed tasks $a_i \times b_{j'}$ (for $b_{j'}$ already on $P_k$)  
- Allocate all unprocessed tasks $a_{i'} \times b_j$ (for $a_{i'}$ already on $P_k$)  

(smaller scheduling overhead: maintain sets of $a$ and $b$ on each processor)
Basic Dynamic Strategies – Comparison

- \( N/l = 100 \) blocks
- Normalized by the lower bound
DynamicOuter: Shortcomings

Limitation:
- When few tasks remains to be processed
- Needs to send many data blocks before reaching these tasks
- Induce useless communication

New strategy: DynamicOuter2Phases
- Start with DynamicOuter
- Switch to the random strategy at the end:
  - Allocate any unprocessed task
  - Send 2 corresponding data blocks
- When the number of unprocessed tasks is below a given threshold
DynamicOuter2Phases: Threshold

- $P = 20$ processors, $N/l = 100$ blocks
- Allows to reduce communications even more
- Optimal threshold: a few percent
- How to determine this threshold?
DynamicOuter2Phases: Analysis

- Assume that the size $N$ of both vectors is large
- Consider a continuous dynamic system with close behavior
- Describe the continuous system using Ordinary Differential Equations
- Approximate the randomized discrete process by the continuous system (no proof)

![Diagram](image)

- Ratio $x = y/N$ of elements of $a$ (and $b$) on $P_k$ at $t_k(x)$
- Basic step: when this ratio goes from $x$ to $x + \delta x = y/N + \ell/N$
- In $\square$: all tasks processed (by $P_k$ or other processors)
Simple Data-Aware Strategy: Analysis

- In $y = xN$:
  - $g_k(x)$ is the fraction of unprocessed tasks (assumed uniformly distributed).
- Time for $P_k$ to compute the red tasks:
  \[
  \frac{2x \delta x g_k(x) N^2}{s_k} = t_k(x + \delta x) - t_k(x)
  \]
- Number of tasks from $τ_k$ computed by other processors during this step:
  \[(t_k(x + \delta x) - t_k(x)) \sum_{i \neq k} s_i\]
- Evolution of $g_k(x)$:
  \[
  g_k(x + \delta x) - g_k(x) = g_k(x)\delta x \frac{-2x\alpha_k}{1 - x^2}
  \]
  where $\alpha_k = \frac{\sum_{i \neq k} s_i}{s_k}$

  \[\Rightarrow g_k(x) = (1 - x^2)^{\alpha_k}\]
Simple Data-Aware Strategy: Analysis

- From $g_k(x)$, it is possible to compute $t_k(x)$, the total communication amount and when to switch to the random strategy.
- Comparison discrete simulation vs. continuous analysis:

\[
\begin{array}{c|c|c|c|c|c}
\hline
\text{Value of } \beta & 2 & 4 & 6 & 8 \\
\hline
\text{Normalized communication amount} & 2 & 2.2 & 2.4 & 2.6 & 2.8 & 3 \\
\hline
\end{array}
\]

- $\beta$: parameter that defines the threshold.
Simulations

Comparison with previous heuristics:

- **β** has a very small deviation with the speed distribution
- Runtime estimation of **β**: use homogeneous speeds
Outline

Introduction

Outer Product
  Static strategies to minimize communications
  Randomized Dynamic Strategies

Matrix Product

Conclusion and Perspectives
**Matrix-Product** \( C = AB \)

- Very similar problem: from 2D to 3D
- Basic computation task: \( C_{i,j} \leftarrow C_{i,j} + A_{i,k}B_{k,j} \)
- Send elements of \( A \) and \( B \), gather elements from \( C \)
- \( A \), \( B \) and \( C \) need to be replicated
- Minimize communication amount
Matrix-Product: Simple Data-Aware Strategy

- Adapt the previous heuristic:
  - $P_k$ knows $xN \times xN$ values of $A$, $B$ and $C$
  - When $P_k$ requests some work
    - Send $2x - 1$ data blocks of $A$, $B$, $C$
    - Allocate all tasks available with these new data

![Diagram of matrix multiplication]

- $xN \times xN$ matrices $A$, $B$, $C$
Adapt the previous heuristic:
- $P_k$ knows $xN \times xN$ values of $A$, $B$ and $C$
- When $P_k$ requests some work
- Send $2x - 1$ data blocks of $A$, $B$, $C$
- Allocate all tasks available with these new data
Adapt the previous heuristic:

- \( P_k \) knows \( xN \times xN \) values of \( A, B \) and \( C \)
- When \( P_k \) requests some work
- Send \( 2x - 1 \) data blocks of \( A, B, C \)
- Allocate all tasks available with these new data
Matrix-Product: Threshold

Similar analysis allows to compute when to switch to the random strategy.

![Graph showing the relationship between normalized communication amount and value of \( \beta \). The graph includes lines labeled DYNAMIC Outer, DYNAMIC Outer 2 Phases, and Analysis, with the y-axis representing normalized communication amount and the x-axis representing the value of \( \beta \).]
Matrix-Product: Simulations

- $N/l = 100$ blocks ($N^3/l^3 = 1,000,000$ tasks)
Outline

Introduction

Outer Product
  Static strategies to minimize communications
  Randomized Dynamic Strategies

Matrix Product

Conclusion and Perspectives
Conclusion and Perspectives

What we have done:
▶ Design and analyze a simple randomized dynamic strategy
▶ Analysis allows to tune the strategy (threshold)
▶ For outer-product and matrix-product (mainly independent tasks)

What we should/will do:
▶ Convergence proof (Mean Field Approximation?)
▶ Analysis in a dynamic environment
▶ Mix static and dynamic strategy
▶ Move to more complex kernels
  ▶ Cholesky decomposition

Thank You!
Conclusion and Perspectives

What we have done:

▶ Design and analyze a simple randomized dynamic strategy
▶ Analysis allows to tune the strategy (threshold)
▶ For outer-product and matrix-product (mainly independent tasks)

What we should/will do:

▶ Convergence proof (Mean Field Approximation?)
▶ Analysis in a dynamic environment
▶ Mix static and dynamic strategy
▶ Move to more complex kernels
  ▶ Cholesky decomposition

Thank You!