Asynchronous iterative algorithms & Reliability

Mourad Hakem

LIFC Laboratory - University of Franche Comté - France

AND Team

Workshop Inria - Illinois 2009 - Paris

Juin 11, 2009

Motivation

- Solve large linear / non-linear equation systems
- Result in problems composed of **millions of components**
- Single computing unit ? — Parallel and distributed platforms

Direct and iterative resolution methods

Equation systems can be solved with direct and iterative methods.

Direct method

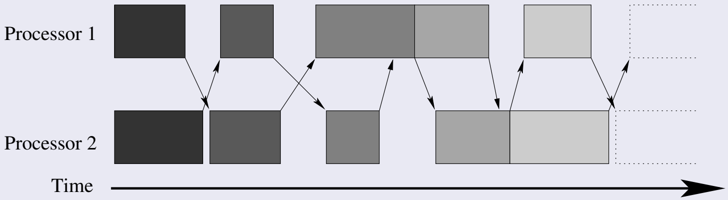
- Gives the **exact solution** for a problem after executing a finite number of operations.
- Does not solve all kinds of problems.
- Not well adapted for large problems.

Iterative method

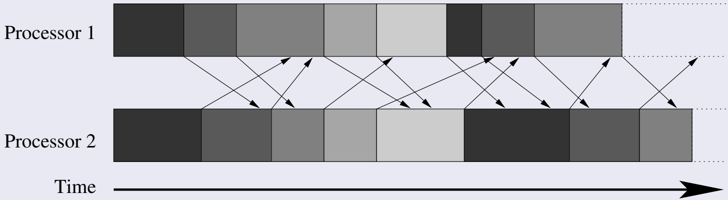
- Iterates until it gives a **good approximation** of the solution.
- Sometimes it is the only way to solve a problem.
- Well adapted for large problems and easy to prallelize.

Iterative methods : Synchronous vs Asynchronous iteration model

Synchronous iteration model



Asynchronous iteration model



Asynchronous iteration model

Advantages

- Eliminates idle times.
- Eliminates synchronizations.
- Tolerates message loss.
- Adapted for heterogeneous and volatile environments.

Difficulties

- Requires a specified asynchronous messaging mechanism.
- Hard to detect the global convergence.

All the difficulties can be overcome with a dedicated platforms:

JACE and JACEP2P-V2

Iterative model

To solve $Ax = b$,

Splitt $A = M - N$, M nonsingular

$$\text{Sequential case:} \quad \begin{cases} given\ x^0 \\ k = 1, 2, ... \\ x^{k+1} = M^{-1}Nx^k + M^{-1}b \\ \equiv x^{k+1} = F(x^k) \end{cases} \qquad (1)$$

Asynchronous iteration model

The asynchronous model is defined by

$$\left\{\begin{array}{l} \textit{Given } X^0 = (X_1^0, ..., X_m^0) \\ \quad \textit{for } k = 1, 2... \\ \qquad \textit{for } i = 1, ..., m \\ \qquad\quad X_i^{k+1} = \left\{\begin{array}{r} F_i(X_1^{r_1^i(k)}, ..., X_m^{r_m^i(k)}) \textit{ if } i \in S(k) \\ X_i^k \textit{ if } i \notin S(k) \end{array}\right. \end{array}\right. \tag{2}$$

- $S(k)$ is the set of components to be updated at step k
- $k - r_j^i(k)$ is the delay of the j^{th} proc.
  The i proc. computing the i^{th} block at the k^{th} iteration
- If $r_j^i(k) = k$ then: synchronous algorithms.

Convergence conditions

Theoritical results

- $\lim_{k\to\infty}\left(M^{-1}N\right)^k = 0 \quad \Leftrightarrow \rho\left(M^{-1}N\right) < 1$
- Synchronous iterations:
  $\rho\left(M^{-1}N\right) < 1 \Rightarrow (1)$ *converges to* $A^{-1}b$
- Asynchronous iterations:
  $\rho\left(\left|M^{-1}N\right|\right) < 1 \Rightarrow (2)$ *converges to* $A^{-1}b$

Motivation

Context

- Heterogeneous platforms - Clusters and Grids
- Iterative applications

Failures?

- Software is assumed to be reliable
- Only hardware failures of nodes
- Faults are assumed to be fail-silent/fail-stop

Fault tolerance objective?

tolerate at most r (reliability factor) node failures

Problem and solutions

Bi-criteria problem

latency & reliability : antagonist objectives

Solutions:

- checkpointing
- task replication
  - Primary/Backup (passive replication)
    - backup is activated only if the fault occurs
    - requires *fault detection/recovery mechanism*
  - Active replication
    - all replicas are activated in parallel
    - no *fault detection/recovery mechanism*

Problem and solutions

Bi-criteria problem

latency & reliability : antagonist objectives

Solutions:

- checkpointing
- task replication
  - Primary/Backup (passive replication)
    - backup is activated only if the fault occurs
    - requires *fault detection/recovery mechanism*
  - Active replication
    - all replicas are activated in parallel
    - no *fault detection/recovery mechanism*

Increasing reliability

- Without replication
  $\mathcal{R}_{\mathcal{A}} = \prod_{i=1}^{n} \mathcal{R}_i$
- With replication

$$\mathcal{R}_{\mathcal{A}} = \prod_{i} \mathcal{R}_i = \prod_{i} \left(1 - \prod_{j} (1 - \mathcal{R}_j)\right), 1 \leq i \leq n, 1 \leq j \leq r+1$$

A brief description of FT-Jace environment

FT-Jace: Reliable Parallel Programming Model for Distributed Computing Environments

Principle

- Multi-threaded
- Convergence detection - Asynchronous iteration model
- Uses the active software *replication scheme* to *mask failures*
- Can tolerate a given number r of arbitrary node failures

A brief description of FT-Jace environment

FT-Jace: Reliable Parallel Programming Model for Distributed Computing Environments

Principle

- Multi-threaded
- Convergence detection - Asynchronous iteration model
- Uses the active software *replication scheme* to *mask failures*
- Can tolerate a given number r of arbitrary node failures

Communication overhead reduction

Communication overhead: $e \rightarrow e(r+1)^2$

– Duplicating each task $r+1$ times is an absolute requirement

– But duplicating each communication $(r+1)^2$ times is not mandatory

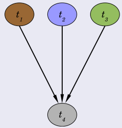
How to reduce communication overhead ?

Communication overhead reduction

All to ALL → One To One mapping ?

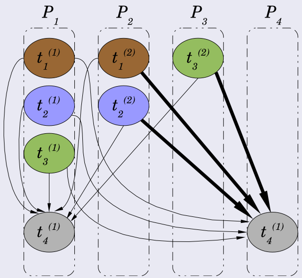
Idea: Try to decrease communication overhead from $e(r+1)^2$ down to at most $e(r+1)$

DAG



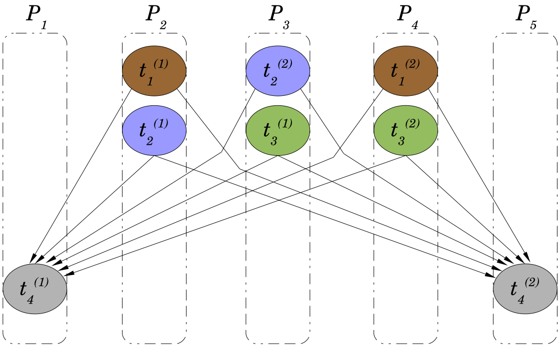
Property 1:

Let t and t_* two tasks involved in communication. If a replica of task t and a replica t_*^z of t_* are mapped on the same node P, then there is no need for other replicas of t_* to send data to node P.



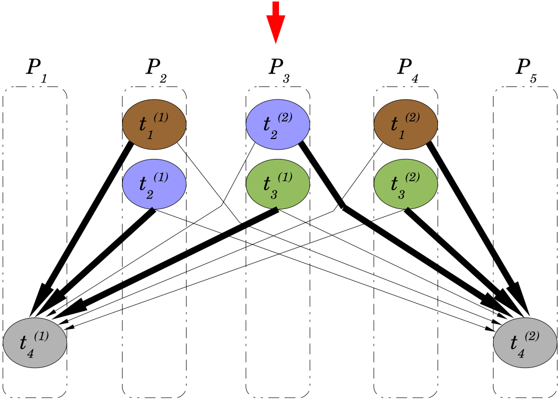
Communication overhead reduction

Counter example



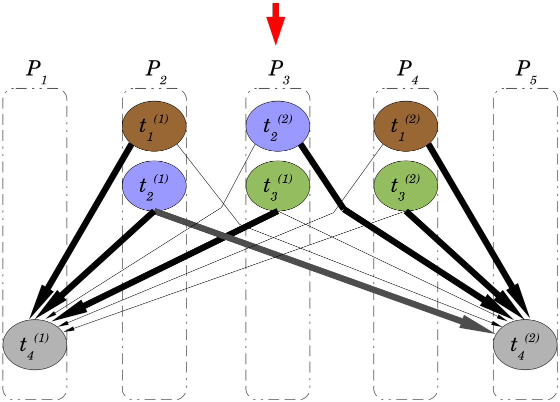
Communication overhead reduction

Counter example



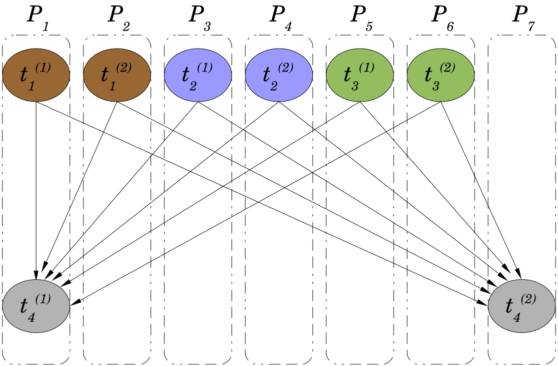
Communication overhead reduction

Counter example



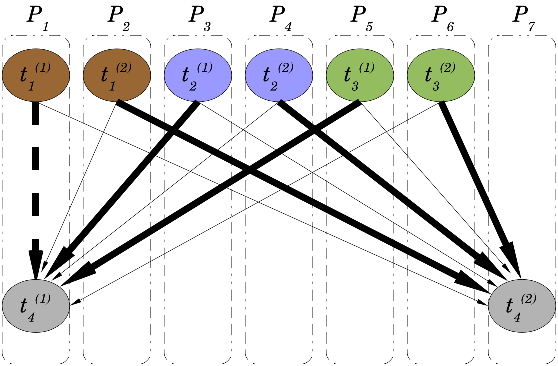
Communication overhead reduction

The number of communications will also be bounded by $e(r+1)$ if at each step replicas are assigned to different processors.



Communication overhead reduction

The number of communications will also be bounded by $e(r+1)$ if at each step replicas are assigned to different processors.



Some theoritical results

Theoritical results (Joint work with A. Benoit and Y. Robert)

- Fork/Outforest graph applications

$$e(r+1)$$

- Classical kernels/Parallel algorithms:
  LU, LAPLACE, STENCIL,DOOLITTLE,LDMt

$$V_2(r+1) + V_3\left(r\left\lceil\frac{(r+2)}{2}\right\rceil + 2\right),\ V_2 \leq \lfloor\frac{e}{2}\rfloor,\ V_3 \leq \lfloor\frac{e}{3}\rfloor$$

- General graph applications:

$$e\left(r\left\lceil\frac{(r+2)}{2}\right\rceil + 1\right)$$

Experimental results

Aim

- Evaluation of FT-Jace performance
- Comparison with fault-free version ($r = 0$)

Parameters

- Frensh Grid'5000 platform
- 50 nodes, bi-core Opteron 2.0 GHZ – Gigabit Ethernet
- Jacobi method to solve linear systems ($Ax = b$)
  (**Synchronous & Asynchronous mode**)
- size of the matrix A is $10^8 \times 10^8$.
- $r = \{1, 2, \ldots, 10\}$

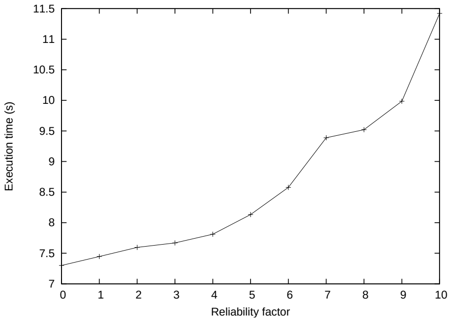
Metrics

- Latency
- Overhead

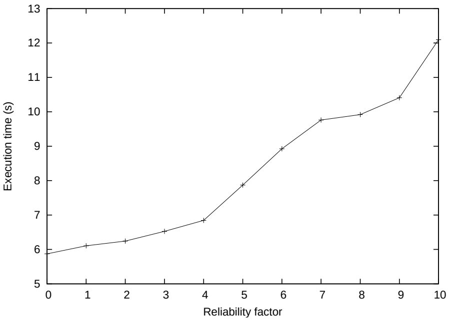
Experimental results

The achieved latency

Asynchronous mode



Synchronous mode



Experimental results

Table: Fault tolerance overhead (%)

Reliability factor (r)	2	4	6	8	10
Asynchronous mode	0.04	0.06	0.17	0.30	0.56
Synchronous mode	0.06	0.16	0.52	0.69	1.06

Fault tolerance overhead increases slightly
as reliability factor goes up

Conclusion

Conclusion

- Iterative computing model
- Asynchronous model is reliable by nature
- Reliability: **FT-Jace** environment
- Good performance

Perspective

- Effective scheduling strategy
- Checkpointing ?