Combining Task- and Data-Parallelism Through Scheduling

Frédéric Suter Frédéric Desprez

IN2P3 Computing Center, CNRS / IN2P3

INRIA - LIP - ENS Lyon

French-US Workshop on Grid Workflows
Sophia Antipolis
March, 23-24 2010

IN2P3 Computing Center in a Few Words

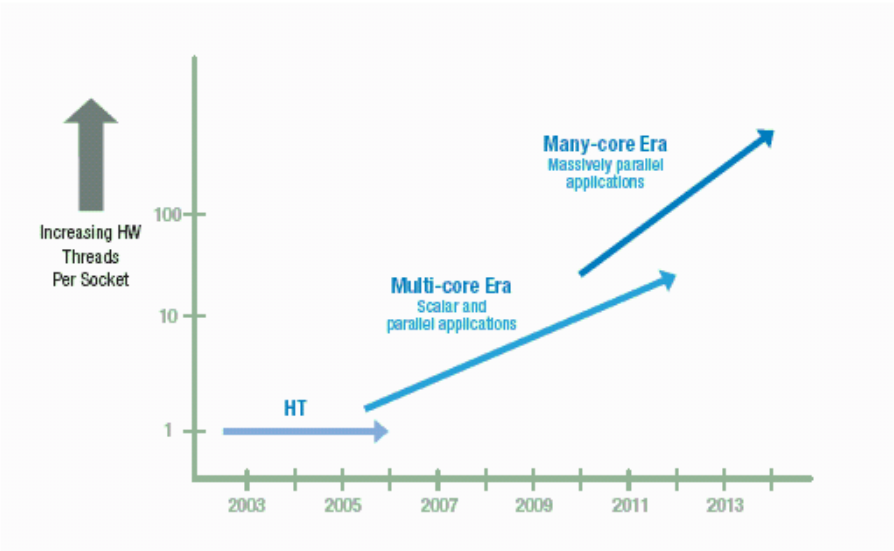
- IN2P3: National Institute of Nuclear Physics and Particule Physics
- Located in Lyon since 1986
- Mostly dedicated to the LHC experiments (Tier-1 center)
- Computing
  - Around 10,000 cores
  - Home made resource manager: Batch Queuing System (BQS)
    - soon replaced by SGE
- Storage
  - On disks and bands (accessed through HPSS)
  - Up to 30 PB of capacity (around 1 PB on disks)
- Network
  - Connected to the French Education Network at 10 GB/sec.
- Research
  - A team since Oct. 2008
  - Goal: establish bridges between research and production grids

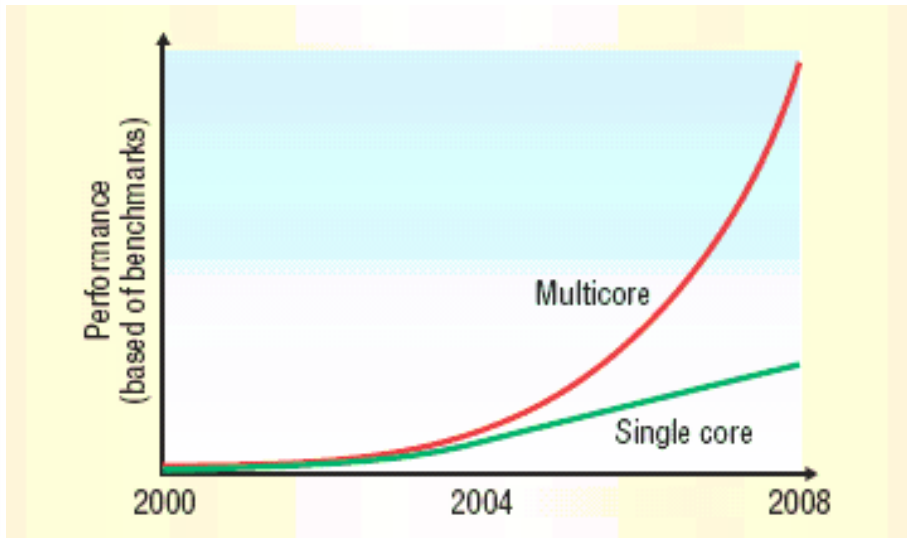
Applications on (Production) Grids

- Many sequential jobs
  - Monte-Carlo simulations
  - Parameter sweep applications
  - Data analysis
- Some parallel jobs
  - MPI or Open-MP programs
- Example of repartition (at IN2P3 Computing Center)
  - 8,600 processors/cores for sequential jobs
  - 1,024 processors/cores for parallel jobs
- More and more workflows
  - Sets of dependent tasks needed to solve a computational problem
  - Advantage: composition of legacy software or services with minimal effort

Evolution of Processor Architectures

- Performance no more coming from clock rate increase
  - Heat dissipation issues
  - High power consumption
- Multicores arising to keep pace with Moore's Law
  - And manycores in a near future





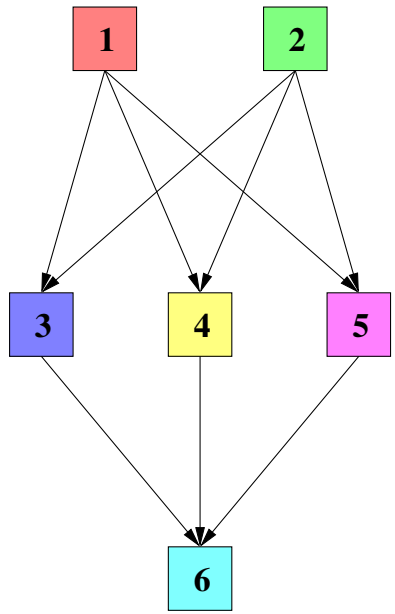
Source: *Platform 2015: Intel Processor and Platform Evolution for the Next Decade, Intel White Paper*

Adequation Between Applications and Processors

- Sequential Jobs
  - Either from parameter sweep or workflow applications
  - Memory becomes the real bottleneck
    - Risks of using 1 core per chip only for some applications
- Parallel Jobs
  - Moving from MPI to more adapted "communication" libraries
- Major update of legacy software
  - Foreseen? Advisable? Mandatory?
  - Similar to the Message Passing revolution some years ago
    - Hard to convince people
    - Necessary to get full performance
  - One example of software evolution: Linear Algebra libraries
    - LAPACK $\rightarrow$ ScaLAPACK $\rightarrow$ PLASMA

Next Generation Workflows?

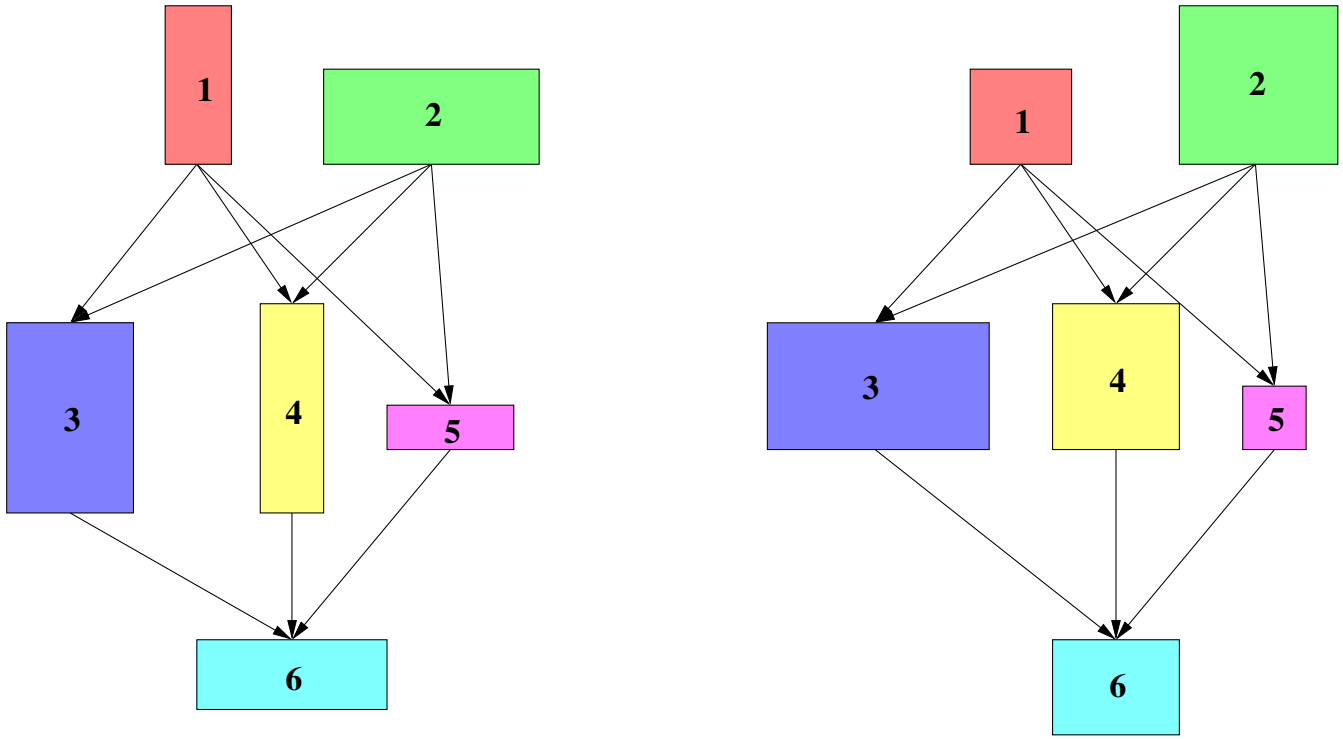
From sequential . . .



- Advantages
  - Keep the task-parallelism of the workflow structure
  - Add data-parallelism in task execution
- Key point
  - Good scheduling algorithms

Next Generation Workflows?

. . . To moldable



- Advantages
  - Keep the task-parallelism of the workflow structure
  - Add data-parallelism in task execution
- Key point
  - Good scheduling algorithms

How to schedule such workflows?

Two steps

1. Determine the right number of processing units per node ⇒ Allocation
2. Find the "right" set of resources to execute each node ⇒ Mapping

Objective functions

- Minimizing the completion time
- Maximizing the efficiency (Are the extra-processors justified?)
- Bi-criteria optimization
- Ensuring fairness (if multiple applications)

Previous Results on that Topic

Single Application

- Homogeneous cluster
  - One-step algorithm: iCASLB
  - Two-steps algorithms: CPA, MCPA, iCPA
  - Extra care on redistribution: RATS
  - Bi-criteria optimization: biCPA
- Heterogeneous multi-clusters
  - Guaranteed algorithm: MCGAS (for almost homogeneous multi-clusters)
  - Adataption of CPA: HCPA
  - Mixed-Parallel HEFT: MHEFT
  - Dynamic application: D-MHEFT

Multiple Applications

- Composing the application graphs
- Constraining the resource allocation: SCRAP
- Coarse grain allocation, Fine grain mapping: CAFM

Previous Results on that Topic

Single Application

- Homogeneous cluster
  - One-step algorithm: iCASLB
  - Two-steps algorithms: CPA, MCPA, iCPA
  - Extra care on redistribution: RATS
  - Bi-criteria optimization: biCPA
- Heterogeneous multi-clusters
  - Guaranteed algorithm: MCGAS (for almost homogeneous multi-clusters)
  - Adataption of CPA: HCPA
  - Mixed-Parallel HEFT: MHEFT
  - Dynamic application: D-MHEFT

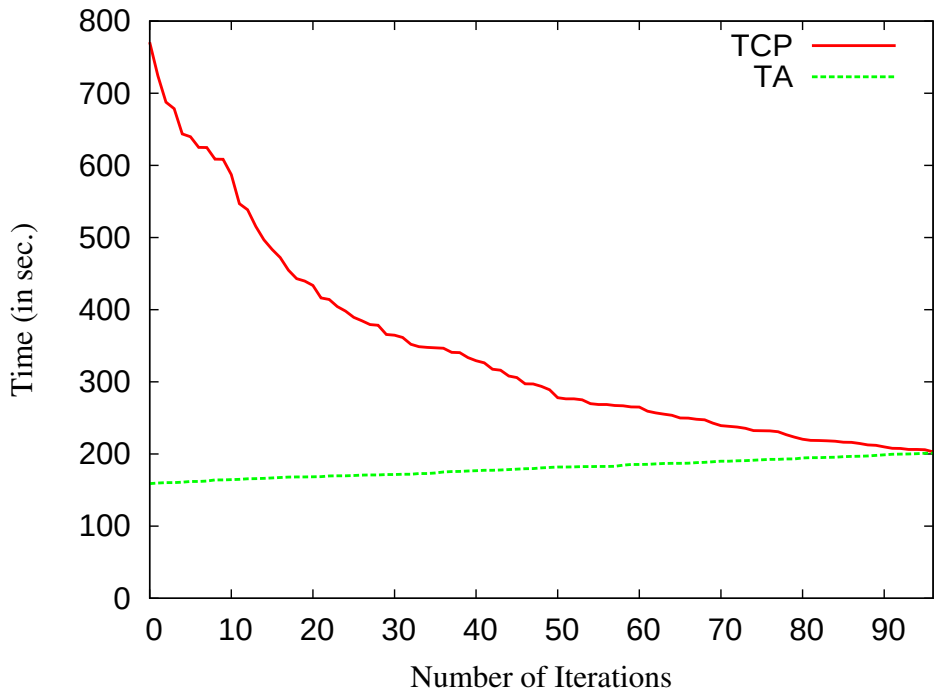
Multiple Applications

- Composing the application graphs
- Constraining the resource allocation: SCRAP
- Coarse grain allocation, Fine grain mapping: CAFM

Principle of CPA and biCPA

- Find an allocation that is a good tradeoff between
  - Makespan: T_{CP}, the critical path length
  - Work: $T_A = \frac{1}{P} \sum_i W(v_i)$, the average area

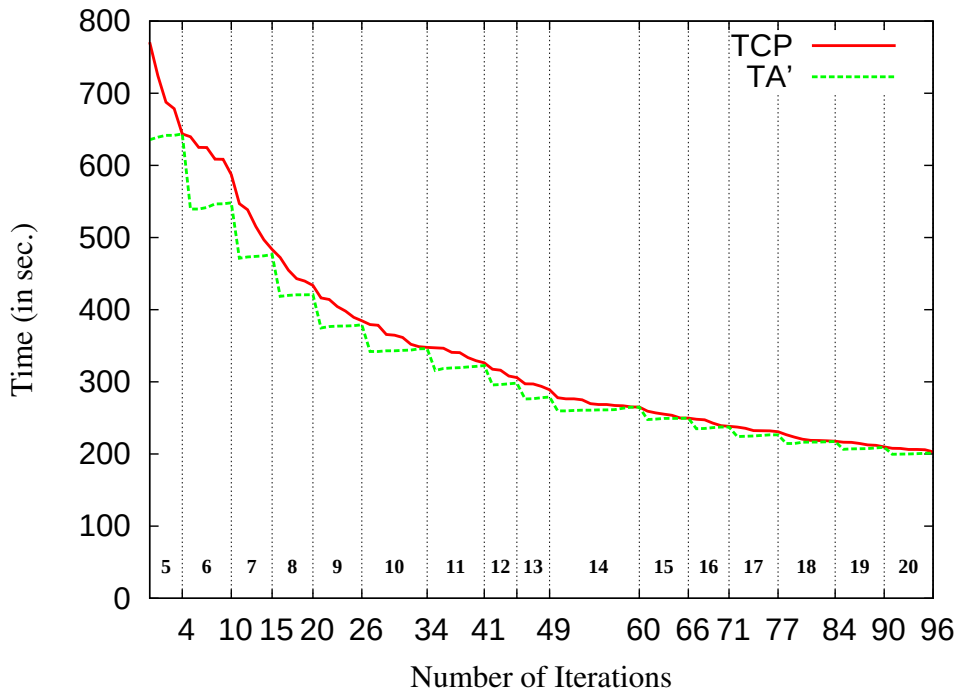
- While ($T_{CP} > T_A$)
  - One extra processor to the most critical task
- Problem
  - Slow convergence when P gets large
- Idea of biCPA
  - Redefine T_A as $\frac{1}{P'} \sum_i W(v_i)$
  - where P' varies from 1 to P



Principle of CPA and biCPA

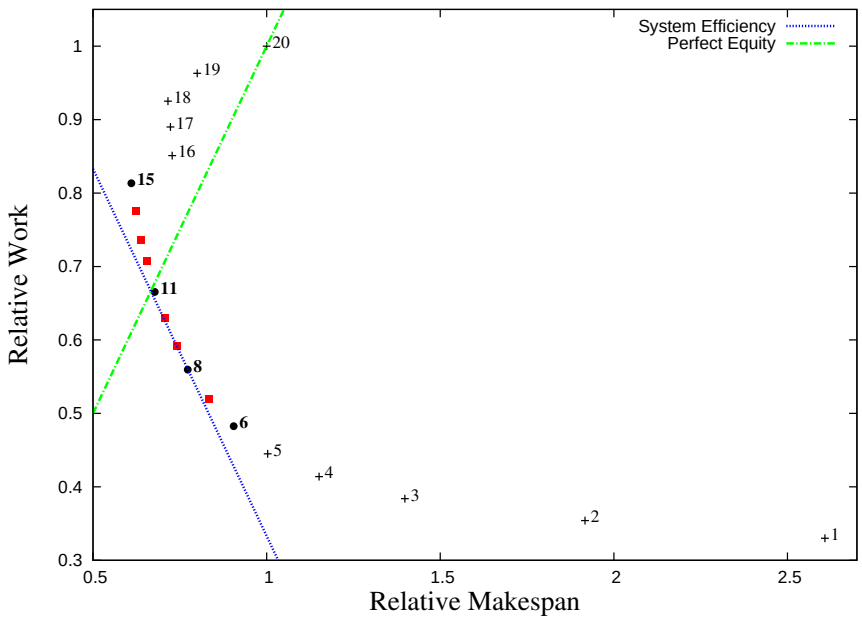
- Find an allocation that is a good tradeoff between
  - Makespan: T_{CP}, the critical path length
  - Work: $T_A = \frac{1}{P} \sum_i W(v_i)$, the average area

- While ($T_{CP} > T_A$)
  - One extra processor to the most critical task
- Problem
  - Slow convergence when P gets large
- Idea of biCPA
  - Redefine T_A as $\frac{1}{P'} \sum_i W(v_i)$
  - where P' varies from 1 to P



Principle of biCPA (Cont'd)

- P distinct allocations
  - For the same complexity
- Which one leads to the best tradeoff between makespan and work?



- Four variants
  - biCPA-M → Minimize Makespan
  - biCPA-W → Minimize Work
  - biCPA-E → Equity
  - biCPA-S → System Efficiency

Frédéric Desprez and Frédéric Suter. A Bi-Criteria Algorithm for Scheduling Parallel Task Graphs on Clusters. In *10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2010)*, Melbourne, Australia, May 2010.

Scheduling a Batch of Moldable Workflows

Performance metrics

- Makespan: user-centric metric
- Average Stretch: average performance as perceived by the jobs
- Maximum Stretch: a fairness metric

If nothing is done

- Each job determine its allocation as if on a dedicated cluster
  - Selfish behavior
- The jobs then compete for resources
  - Can be harmful for small jobs

What can be done?

- Give an higher priority to small jobs
- Merge the jobs
- Force the jobs to build their allocation on a subset of the cluster
- Do some bin packing of independent moldable jobs

Sketch of the CAFM-K-Shelves Algorithm

1. Determine the makespan of each job on each number of processors
   - Get a moldable profile of each job
2. Partition the time in K phases, or "shelves"
   - Depends on an approximation of the optimal makespan and smallest execution time
3. For each shelf
   - Solve a knapsack problem
   - Maximize the work executed in the current shelf
   - Determine an coarse grain allocation for each moldable job
   - Schedule the "boxes" representing each selected job
   - Schedule each task graph wihtin its box (fine grain mapping)
4. Open the boxes and do some backfilling

Henri Casanova, Frédéric Desprez and Frédéric Suter. On Cluster Resource Allocation for Multiple Parallel Task Graphs. Submitted to *Journal of Parallel and Distributed Computing*.

Also available as INRIA Research Report RR-7224.