Dynamic Fractional Resource Scheduling Practical Issues and Future Directions

Mark Stillwell Frédéric Vivien Henri Casanova

INRIA, Université de Lyon, LIP

International Research Workshop on Advanced High Performance Computing Systems
Cetraro, Italy
June 27–29, 2011

Outline

Outline

Dynamic Fractional Resource Scheduling

- HPC Scheduling Problem:
  - homogeneous nodes with high-speed interconnect
  - network file system
  - release date and number of tasks known at submission
  - computation time known only after completion
  - goal: minimize maximum stretch
- approach:
  - based on virtual machine technology
    - fractional resource allocations
    - preemption / migration
  - on-line scheduling problem $\rightarrow$ sequence of off-line resource allocation problems
  - computable, on-line performance metric related to max stretch
  - heuristics for task placement/resource allocation

Task Placement and Resource Allocation Problem

- nodes comprise a number of resources (CPU cores, memory, bandwith, etc...)
- tasks have rigid requirements and fluid needs
  - job tasks have the same requirements and needs
  - rigid requirements specify minimum resource allocations
  - fluid needs specify minimum allocations without performance degradation
- yield of a job is a value between 0 and 1
  - correlated with performance
  - jobs allocated product of yield and need in each resource
- Goal: find an allocation that maximizes the minimum yield

Task Placement Heuristics

- **Greedy Task Placement** – incremental, puts each task on the node with the lowest computational load on which it can fit without violating memory constraints
- **MCB Task Placement** – global, iteratively applies multi-capacity (vector) bin-packing heuristics during a binary search for the maximized minimum yield
  - achieves higher minimum yield values than Greedy
  - can *potentially* cause lots of migration
- what if the system is oversubscribed?
  - use a priority function to decide which jobs to run

Priority Function?

- Virtual Time: subjective time experienced by a job
- first idea: $\frac{1}{\text{VIRTUAL TIME}}$
  - informed by ideas about fairness
  - lead to good results
  - theoretically prone to starvation
- second idea: $\frac{\text{FLOW TIME}}{\text{VIRTUAL TIME}}$
  - addresses starvation problem
  - lead to poor performance
- Third Idea: $\frac{\text{FLOW TIME}}{(\text{VIRTUAL TIME})^2}$
  - combines idea #1 and idea #2
  - addresses starvation
  - performs about the same as first priority function

When to apply Heuristics

We consider a number of different options:

- Job Submission – heuristics can use greedy or bin packing approaches
- Job Completion – as above, can help with throughput when there are lots of short running jobs
- Periodically – some heuristics periodically apply vector packing to improve overall job placement

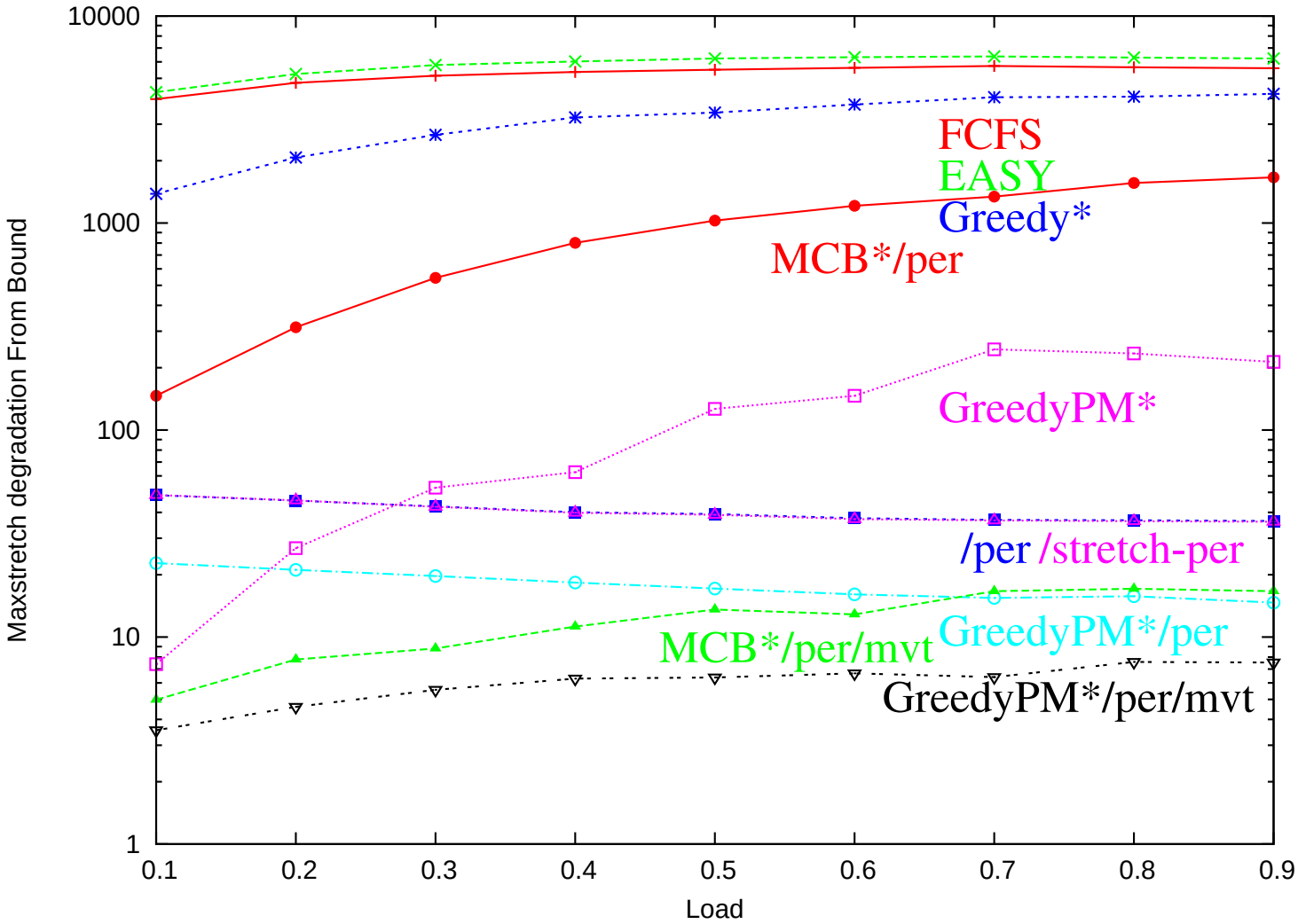
Methodology

- experiments conducted using discrete event simulator
- mix of synthetic and real trace data
- considered only memory requirements and CPU needs
- preemption/migration of jobs causes 300 second penalty to runtime
- periodic approaches use a 600 second (10 minute) period
- absolute bound on max stretch computed for each instance
- performance comparison based on degradation from max stretch bound

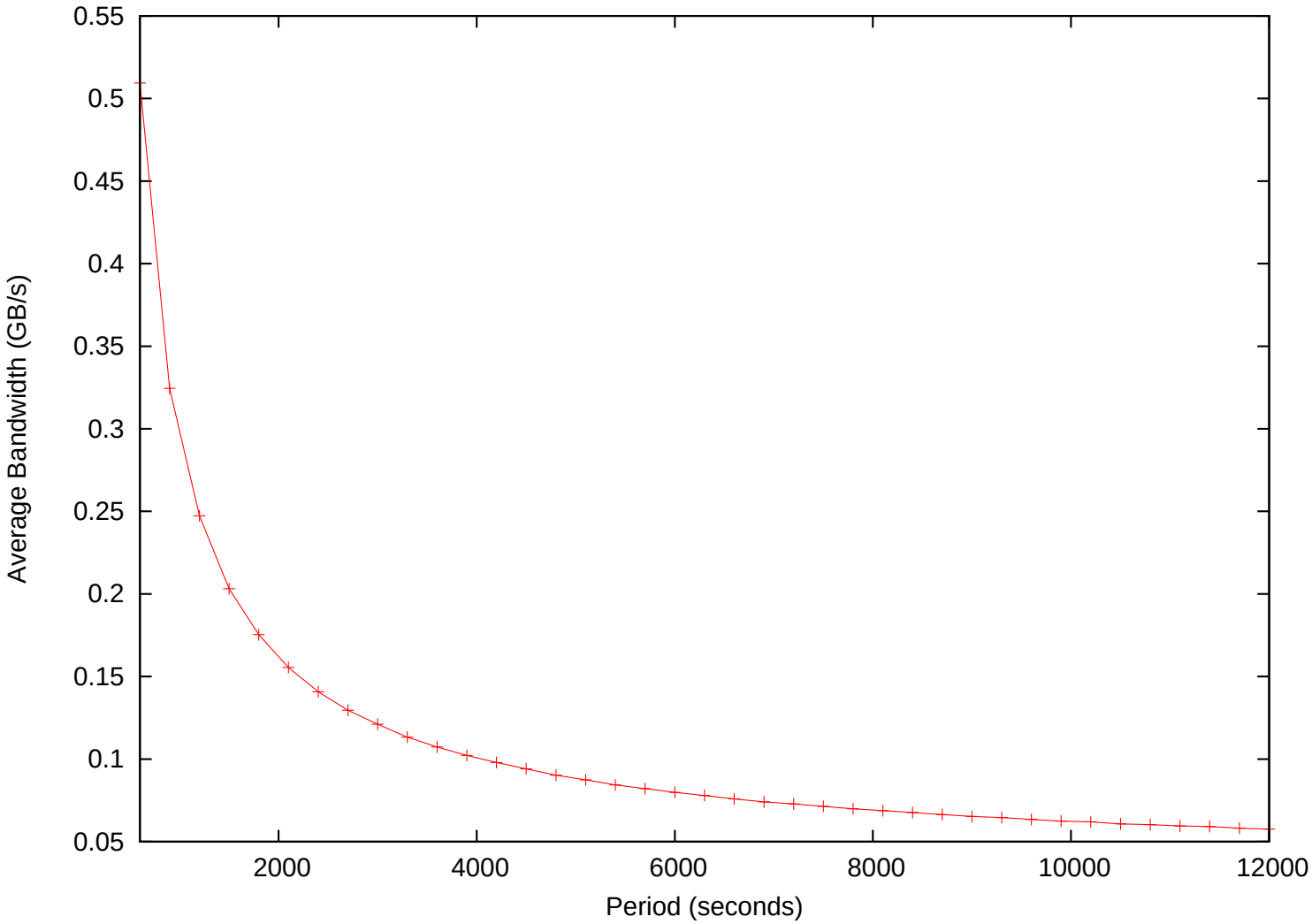
Simulation Algorithms

- FCFS
- EASY
- Greedy*
- GreedyPM*
- /per
- GreedyPM*/per
- MCB*/per
- GreedyPM*/per/mvt
- MCB*/per/mvt
- /stretch-per

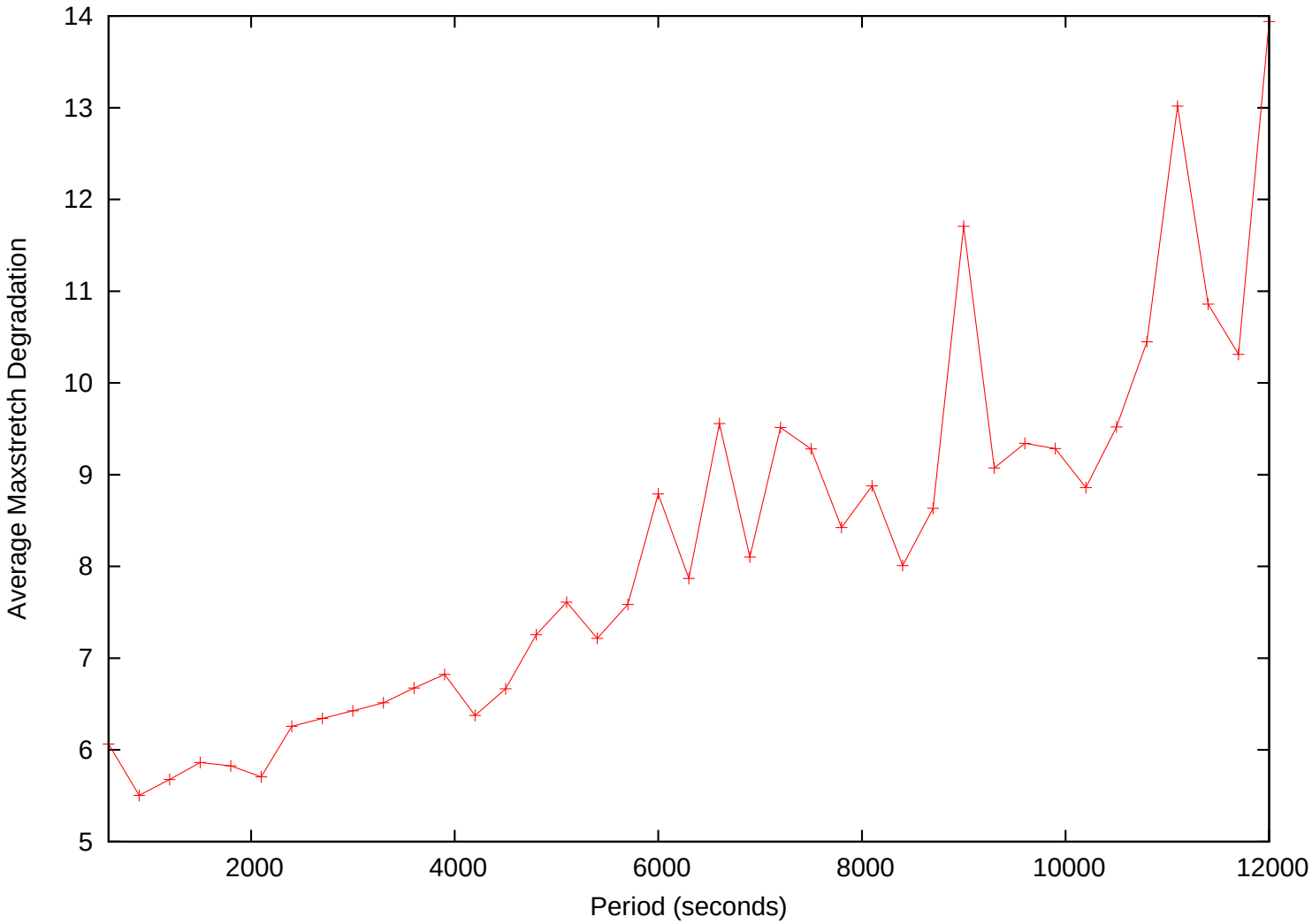
Max Stretch Degradation vs. Load, 5 minute penalty



Bandwidth vs. Period



Max Stretch Degradation vs. Period



Outline

Development Platform

- Grid5000 (`http://www.grid5000.fr`)
  - Kadeploy images
  - Genepi Nodes: Quad Core Xeon 2.5GHz w/ 8GB ram
- Open-Source Software
  - Xen 4.0
  - Debian Squeeze / Linux 2.6.32

Goals

- “small” experiment topics:
  - VM overhead
  - resource requirement/needs estimation
  - time sharing performance impact
  - preemption/migration costs
    - bandwidth consumption
    - performance impact
  - migration planning strategies
- “large” experiment topic: DFRS in practice

Goals

- “small” experiment topics:
  - VM overhead
  - resource requirement/needs estimation
  - time sharing performance impact
  - preemption/migration costs
    - bandwidth consumption
    - performance impact
  - migration planning strategies
- “large” experiment topic: DFRS in practice

Goals

- “small” experiment topics:
  - VM overhead
  - resource requirement/needs estimation
  - time sharing performance impact
  - preemption/migration costs
    - bandwidth consumption
    - performance impact
  - migration planning strategies
- “large” experiment topic: DFRS in practice

Goals

- “small” experiment topics:
  - VM overhead
  - resource requirement/needs estimation
  - time sharing performance impact
  - preemption/migration costs
    - bandwidth consumption
    - performance impact
  - migration planning strategies
- “large” experiment topic: DFRS in practice

Goals

- “small” experiment topics:
  - VM overhead
  - resource requirement/needs estimation
  - time sharing performance impact
  - preemption/migration costs
    - bandwidth consumption
    - performance impact
  - migration planning strategies
- “large” experiment topic: DFRS in practice

Estimating Resource Needs

- Techniques:
  - repeated runs for benchmarking / model building
  - online model building
  - introspection / configuration variation
- Models:
  - constant values
  - time-varying values
  - random variables / probability distributions
- Questions:
  - how accurate can we be?
  - how accurate do we need to be?

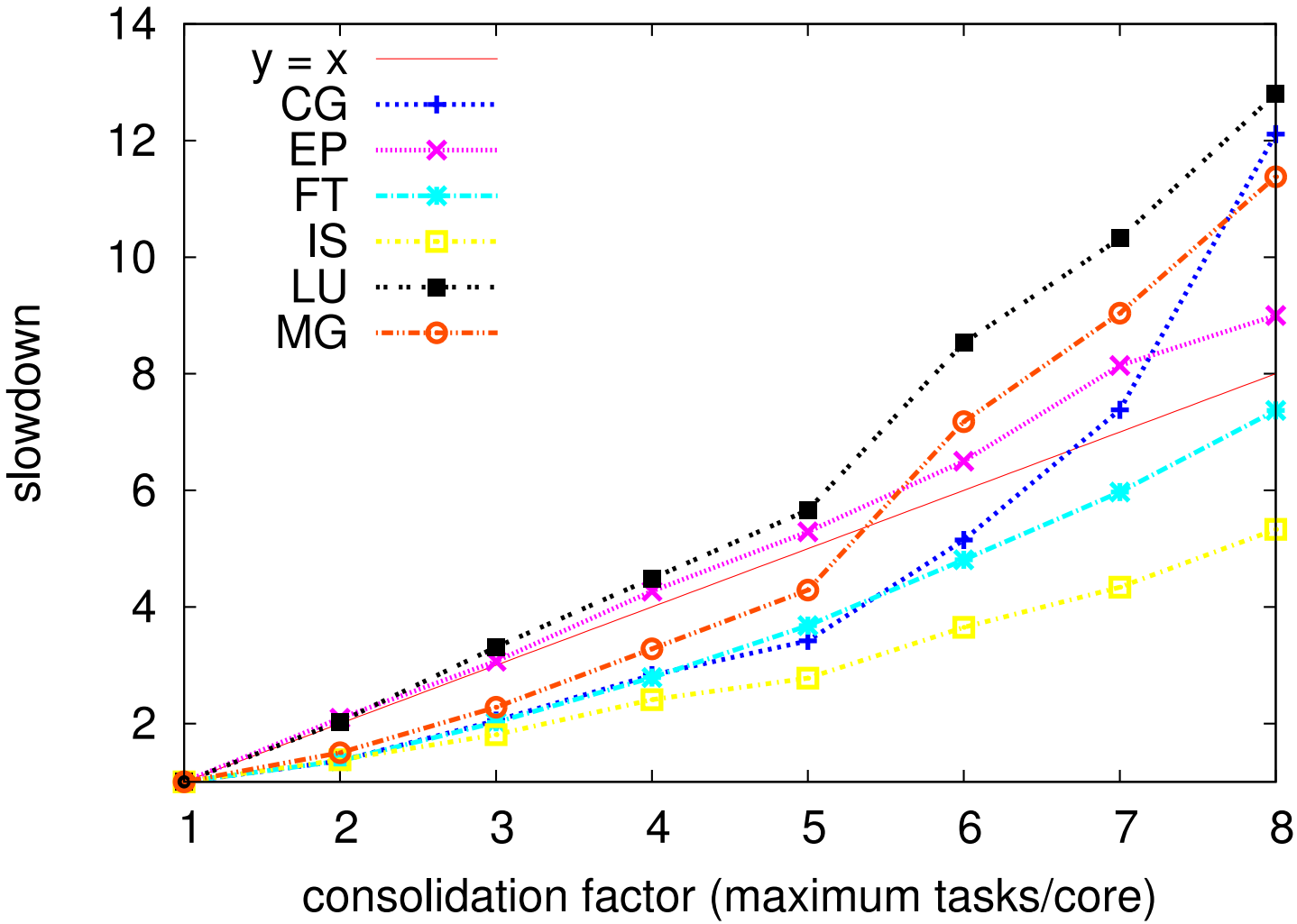
Benchmark Task VM Memory Requirements

bench/tasks	1	2	3	4
CG	477M	251M	–	147M
EP	<64M	<64M	<64M	<64M
FT	1930M	977M	–	499M
IS	637M	329M	–	176M
LU	218M	125M	–	80M
MG	485M	259M	–	147M
SP	369M	–	–	123M

Time Sharing Performance Impact

- Process thrashing!!!
  - Gang Scheduling
  - Flexible Co-scheduling
- Uncoordinated might not be so bad...

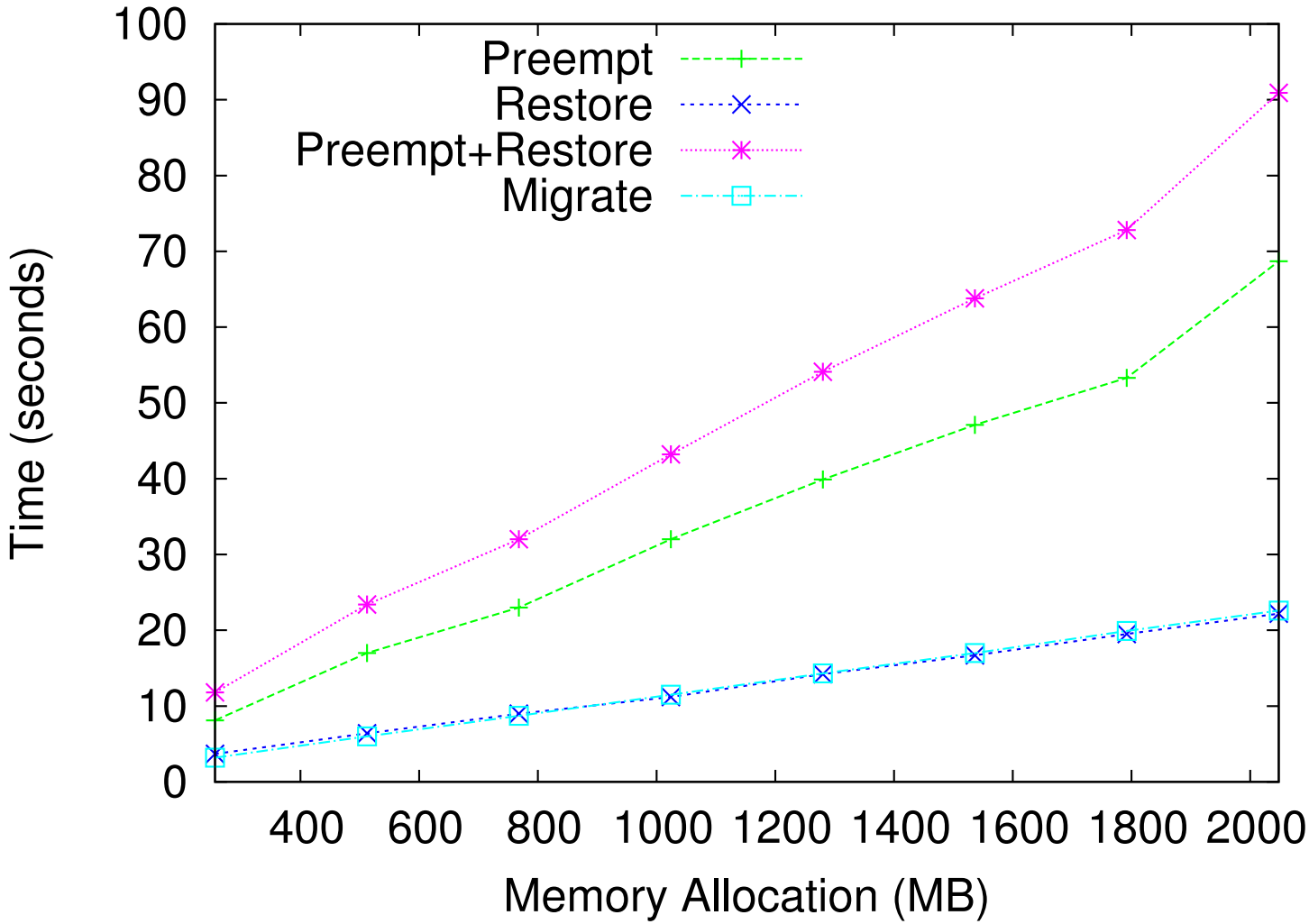
VM Consolidation



Parallel Task Preemption/Migration

- preemption of parallel distributed jobs is theoretically difficult
  - in the general case this may require coordinated shutdown
  - on the target platform everything happens quickly enough that it isn't an issue
- migration can be done by preempt/restore or "live"

Preemption/Migration Timing Results



Migration Planning

- using only live migration is NP-hard
- could fail (circular dependencies)
- heuristic: try to move the highest priority job
  - if it can't move, try next highest
  - if none can move, preempt the lowest priority job scheduled for migration and try again
  - restore any preempted jobs at the end

Planned Implementation Validation Experiment

- generate traces using an accepted model
- annotate them with CPU/memory values
- ”fill in” the traces using benchmarks...
  - NAS parallel benchmarks (HPC)
  - TCPP benchmarks (Service Hosting)

Future Work

- complete development of practical implementation
- extend the model
  - nonlocal resources
  - heterogeneous nodes
  - varying communications delays
- new or arbitrary optimization targets
  - fault tolerance
  - energy or other costs
  - formal definitions of fairness

Summary

- DFRS performs well in simulation
  - orders of magnitude better than batch
  - close to a theoretical bound
- prototype in development
  - preliminary results are promising

References I


Stillwell, M., Schanzenbach, D., Vivien, F., and Casanova, H. (2010).
Resource allocation algorithms for virtualized service hosting platforms.
Journal of Parallel and Distributed Computing, 70(9):962–974.


Stillwell, M., Vivien, F., and Casanova, H. (2011).
Dynamic fractional resource scheduling vs. batch scheduling.
IEEE Transactions on Parallel and Distributed Systems. accepted for publication.