Universitat Autonoma de Barcelona

Performance Models for Dynamic Tuning of Parallel Applications on Computational Grids

Genaro Costa, Josep Jorba, Anna Morajko, Tomas Margalef and Emilio Luque

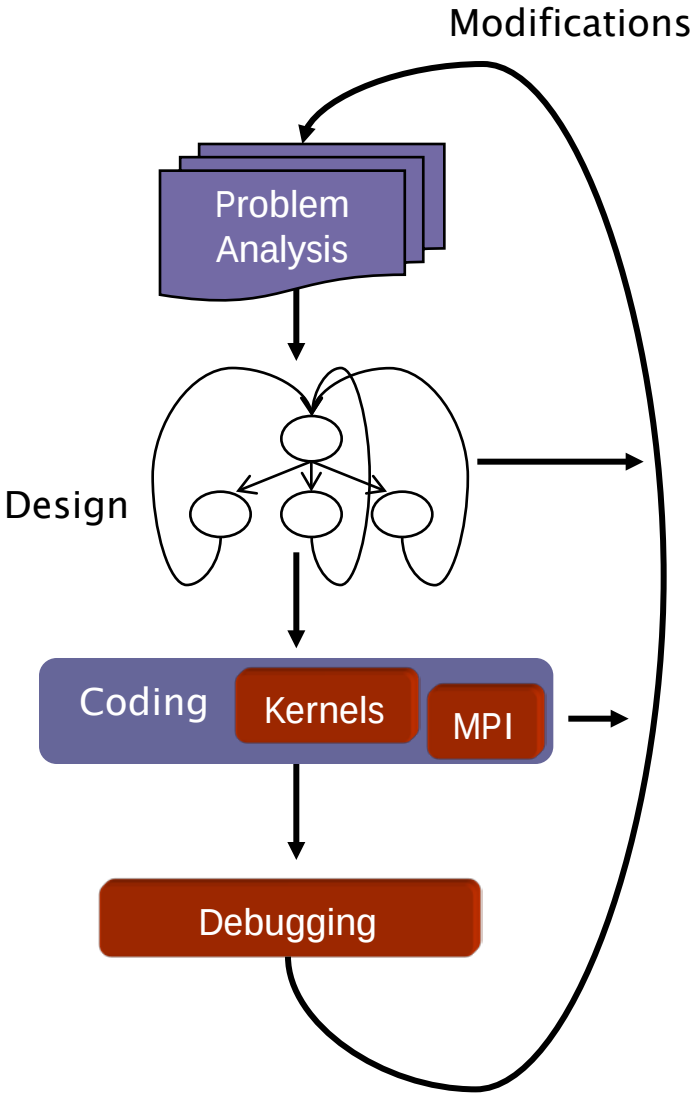
Cluster 2008

Content

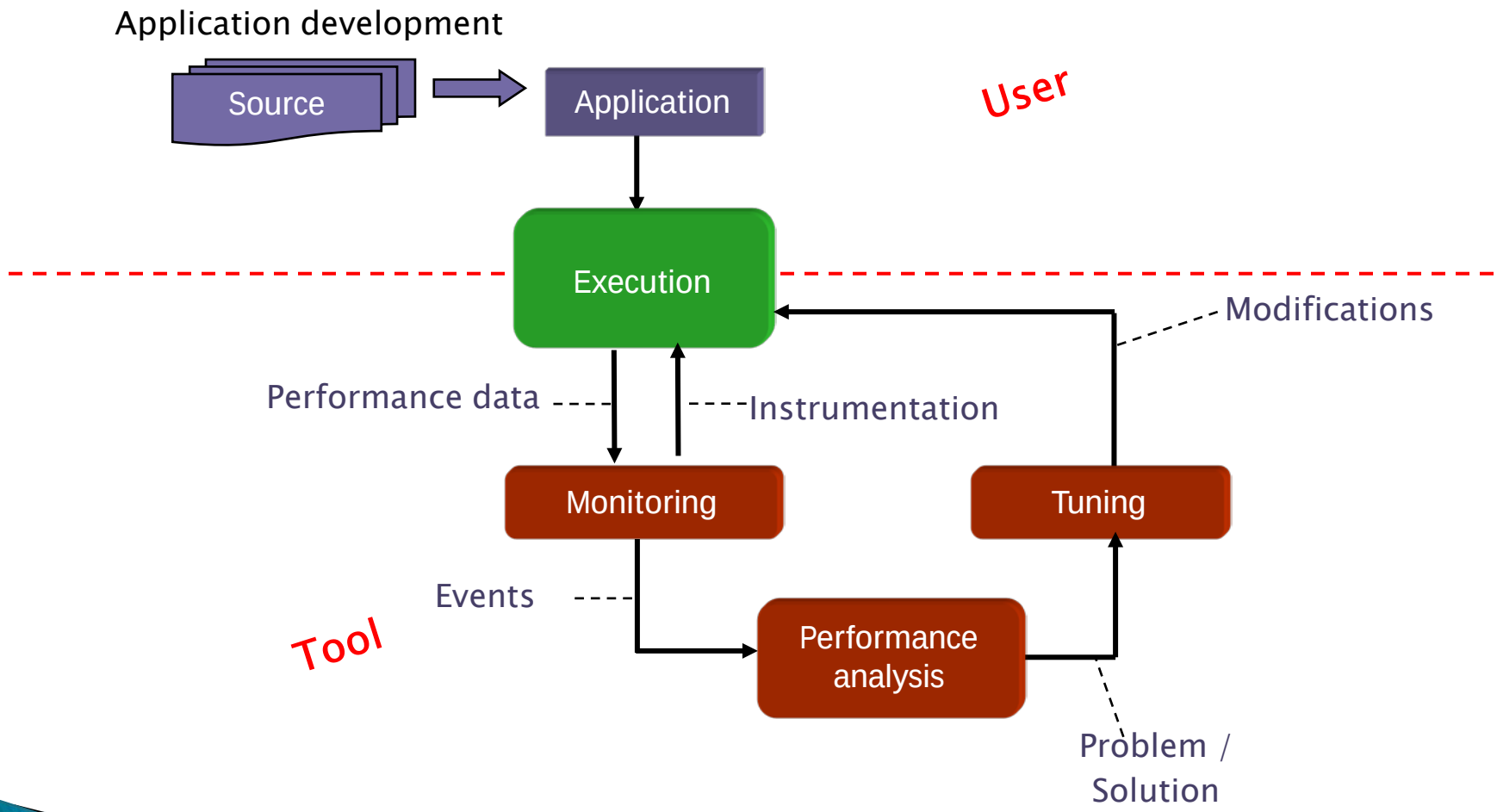
1. Motivation
2. Performance Model for Dynamic Tuning
3. Tuning Heuristics
4. Experimentation Results
5. Conclusions

Motivation

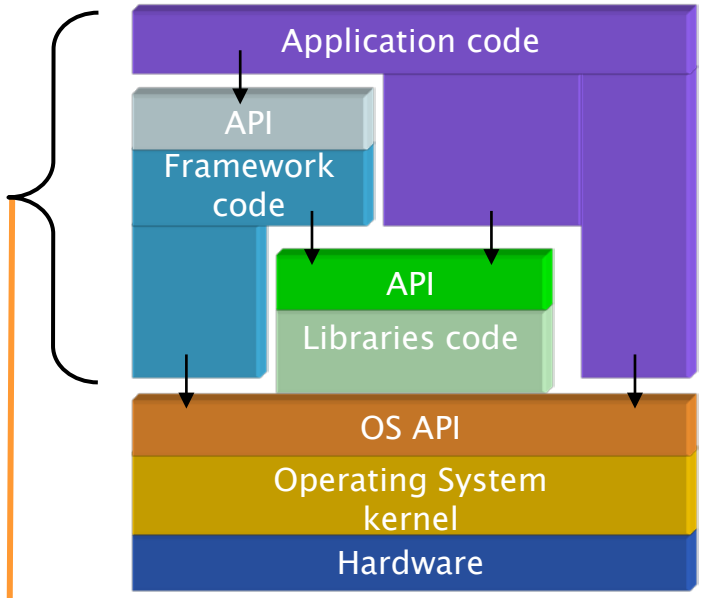
- Performance is a main issue in parallel application development.
- Runtime variation of system characteristics harms application performance.
- Computational Grids are heterogeneous and dynamic by nature.
- Dynamic tuning may help application increasing its execution performance indexes.



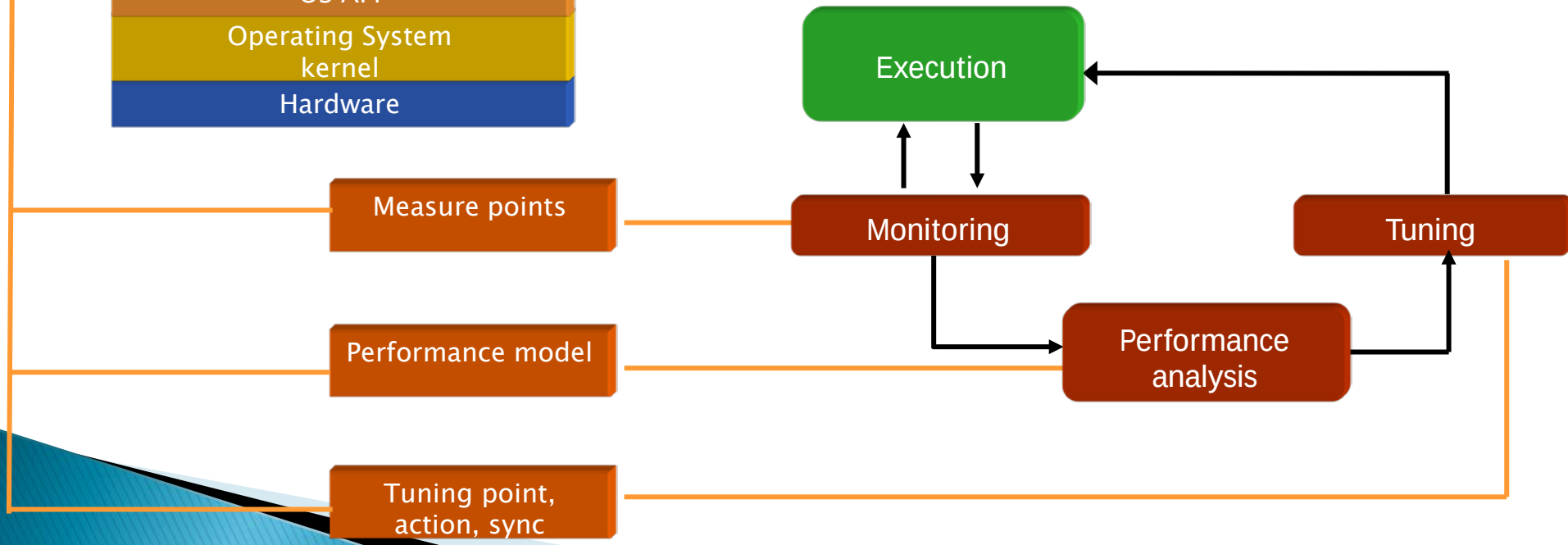
Automatic Performance Tuning



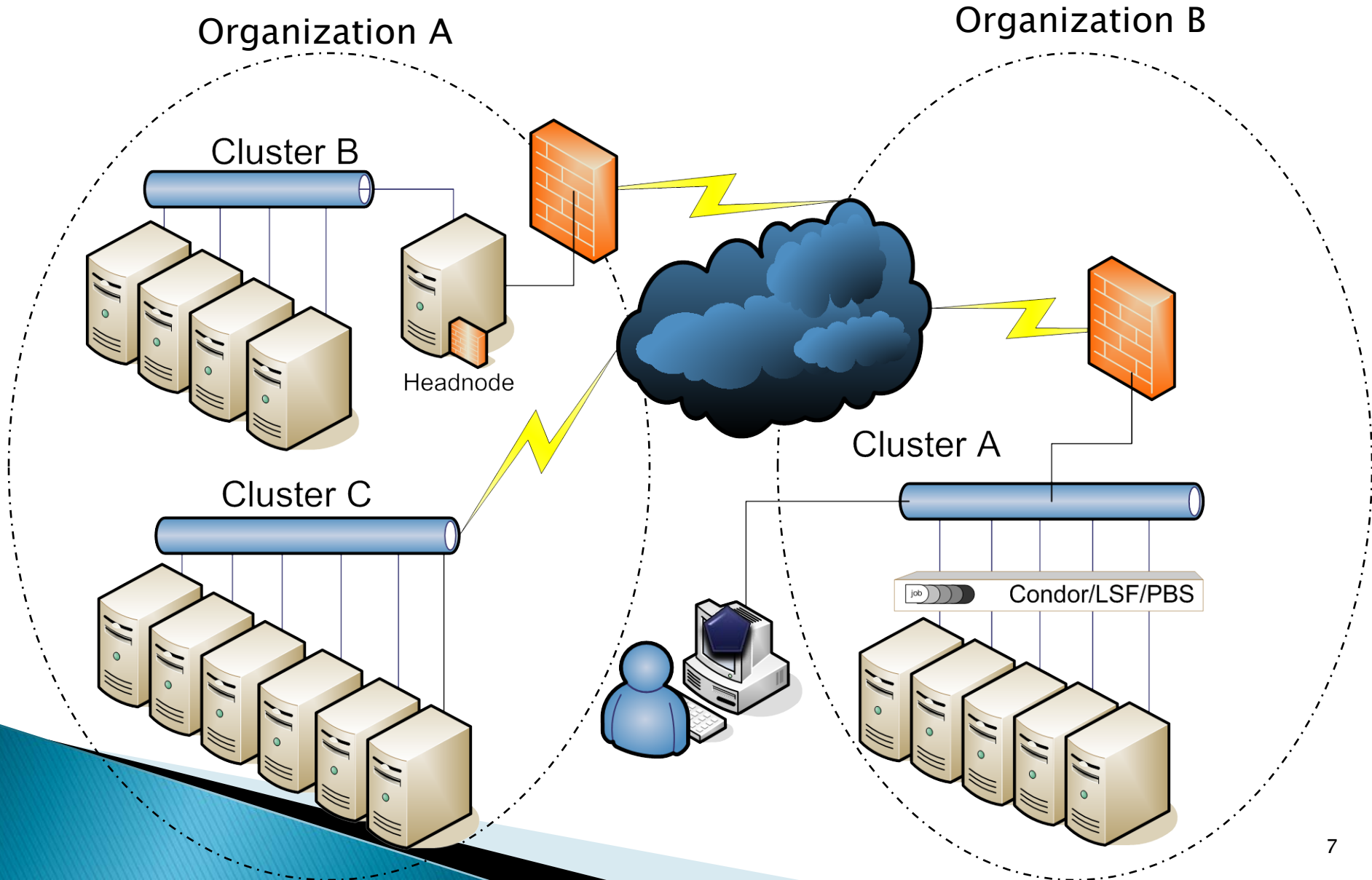
Dynamic Performance Tuning



Performance Model Driven Tuning process



Computational Grids



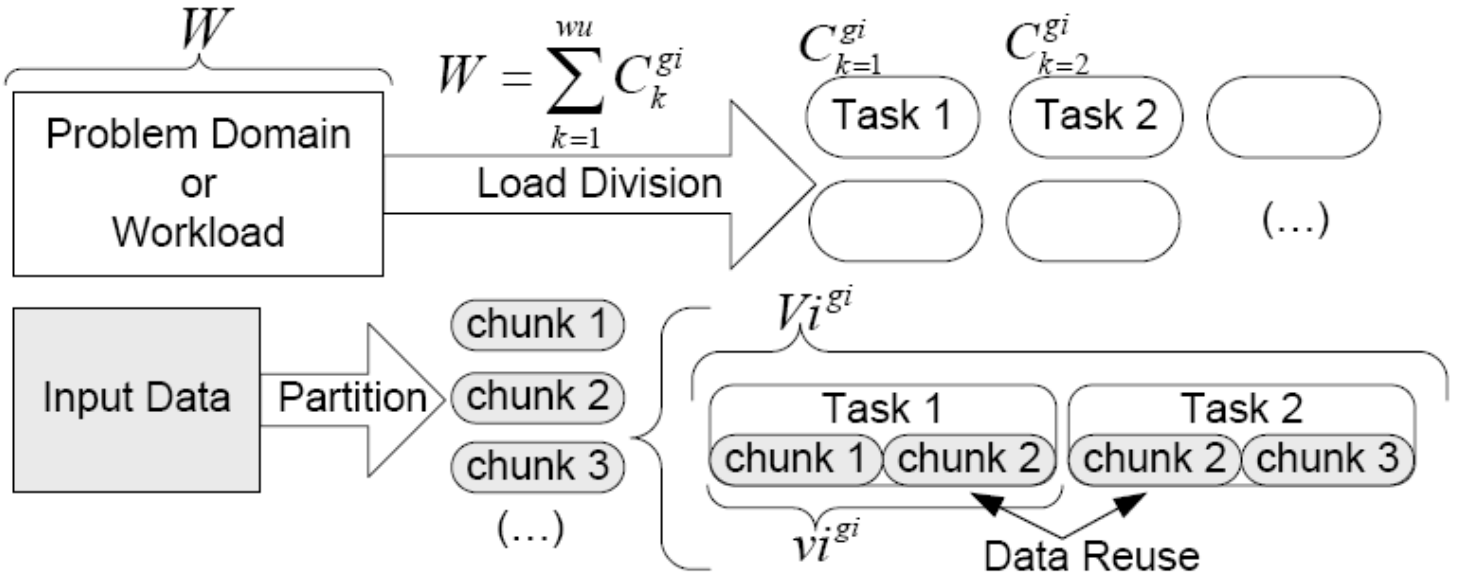
Content

1. Motivation
2. Performance Model for Dynamic Tuning
3. Tuning Heuristics
4. Experimentation Results
5. Conclusions

Assumptions

- Tuning on frameworks provides a balance between performance improvement and generalization.
- Master-Worker programming helps application deal with system heterogeneity.
- The significant parameters for Master-Worker tuning is Number of Workers and Grain Size (Compute/Computation ration) selection.
- Application should support change of number of workers and grain size selection at runtime.

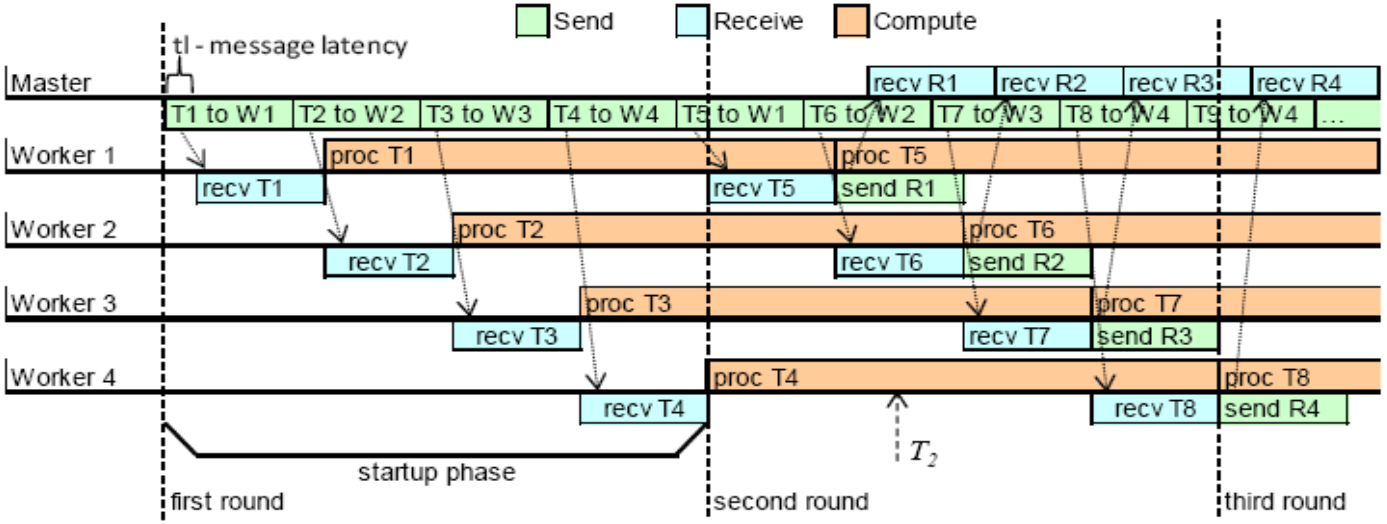
Changing Grain Size (Compute/Communication Ratio)

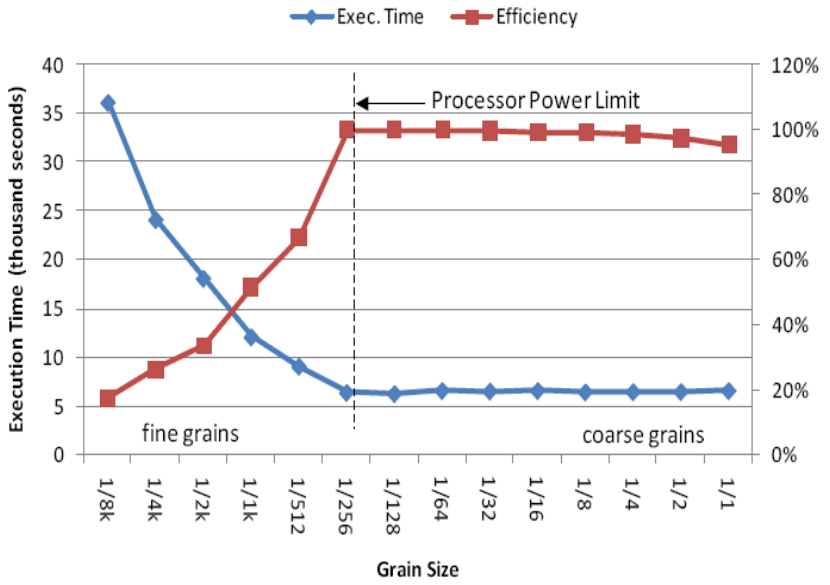


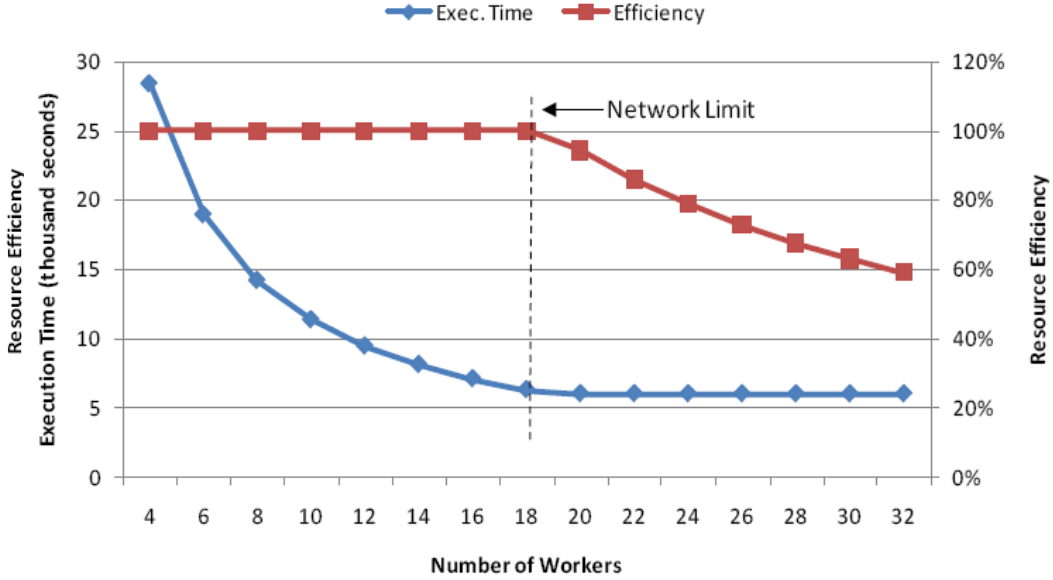
- Smaller Tasks → Higher communication volume
- Coarser Tasks → Lower communication volume

Grain Size Selection changes Compute/Communication Ratio

Master/Worker Applications







Performance Indexes

- Scenario and Problem
  - Given a set of resources assigned and mapped by a meta-scheduler/resource broker, how to reduce application execution time and increase resource usage efficiency?
- Objectives
  - Lower total execution. ← Grain size selection.
  - Increase resource usage efficiency. ← Number of Workers.

Performance Models

- The minimal execution time is achieved when application have the maximum number of faster workers in busy state.
  - Grain Size selection helps application to use the assigned resource set.

$$N_{opt}^{gi} = \frac{Tc^{gi}}{\max\left(vi^{gi}, vo^{gi}\right) * \lambda}$$

Direct impact on resource usage efficiency

$$T_{startup}^{gi} = tl + \left(vi^{gi} * \lambda\right) * \left(\frac{N_w + 1}{2}\right)$$

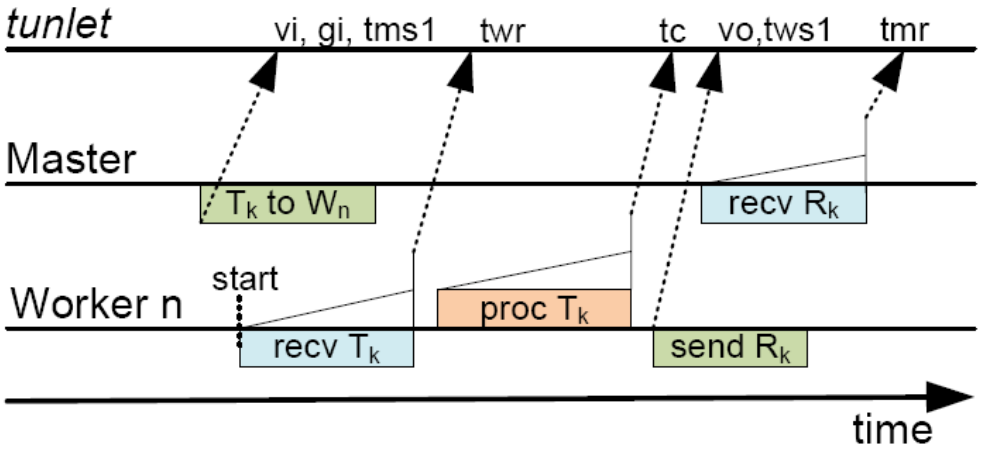
$$T_{finalization}^{gi} = tl + \left(vo^{gi} * \lambda\right) * \left(\frac{N_w + 1}{2}\right)$$

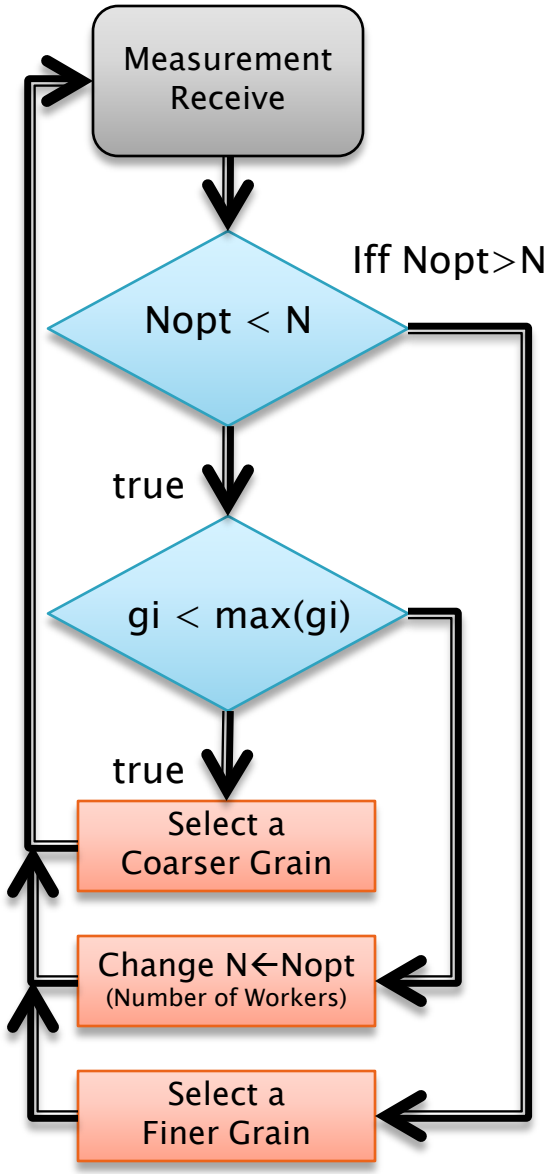
Content

1. Motivation
2. Performance Model for Dynamic Tuning
3. Tuning Heuristics
4. Experimentation Results
5. Conclusions

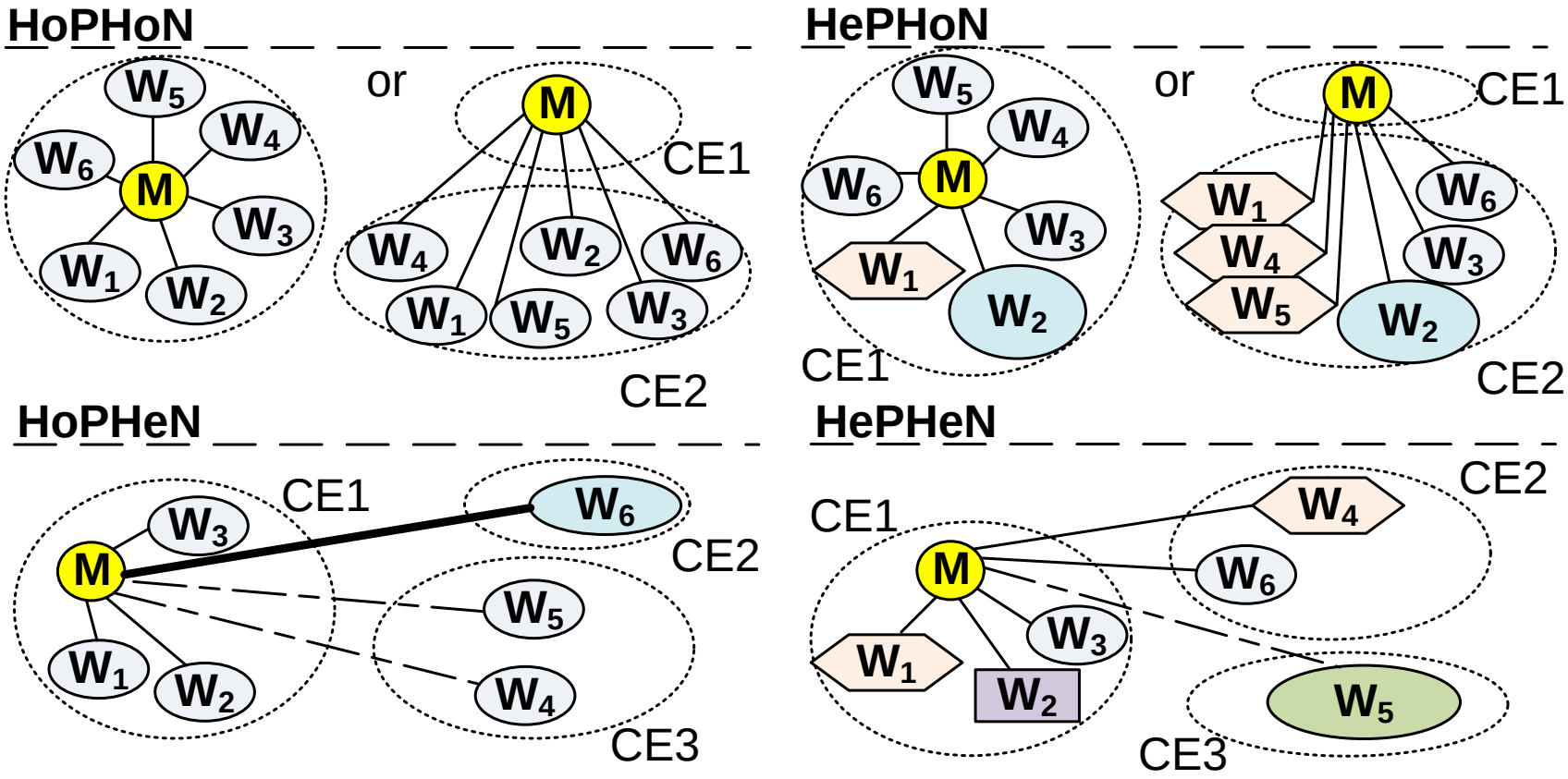
Tuning Process

- Runtime compute and communication of each task per worker are continuously measured.





Processor and Network Heterogeneity



Heterogeneous Scenarios

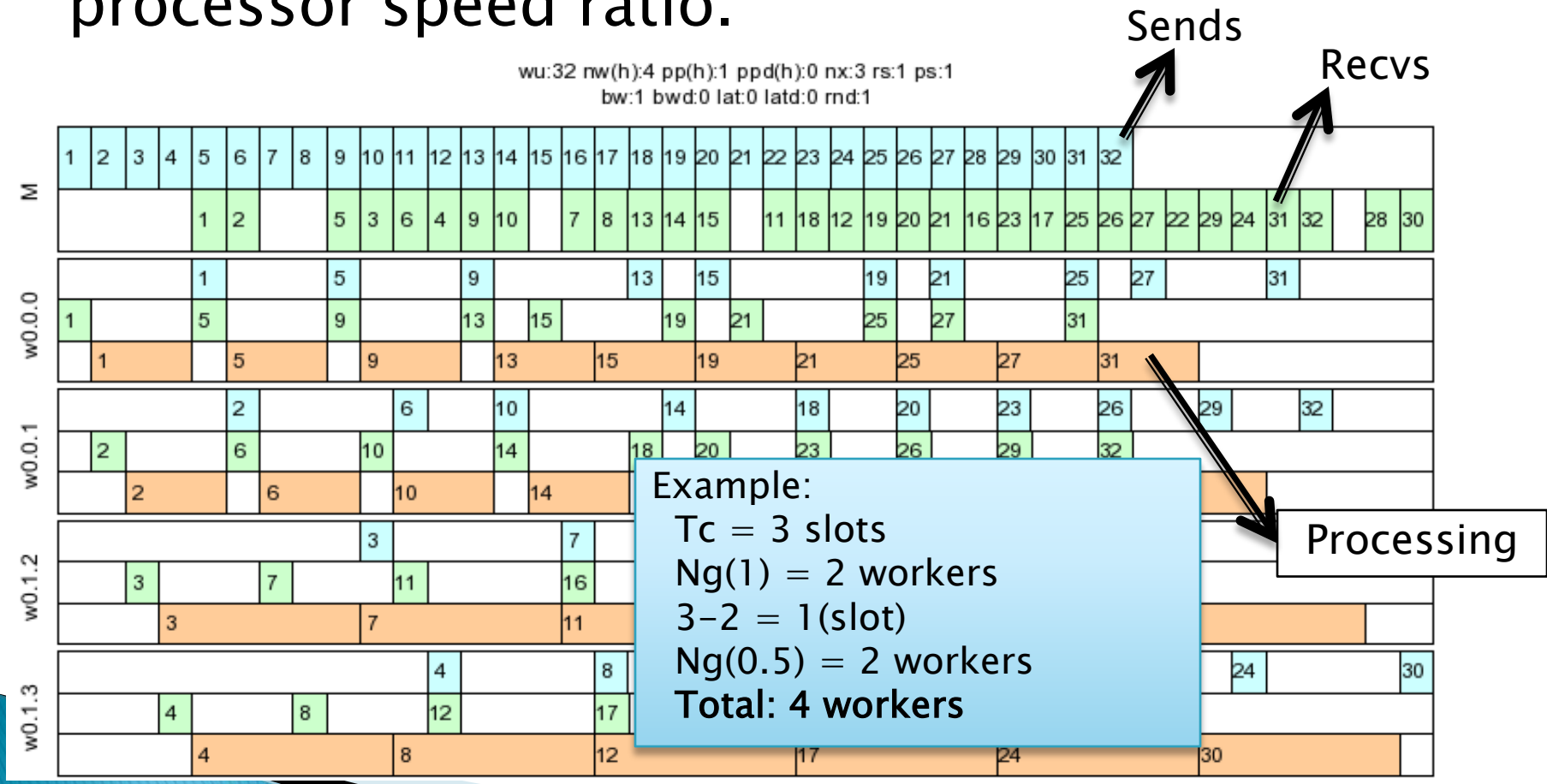
- Task distribution gives higher priority to faster processors.
- Remote Compute Hosts (CH) are grouped by Compute Element (CE) witch receives tasks in parallel.
  - Communication managers isolates master to remote worker communication blocking.
- The number of workers in each CE is independently tuned (greedy approach).

Tune Heuristic

```
let slot = 0 and nw = 0
for each Wg(k) from k=0 to Wgn-1
   let Nopi = calc(Nopt for Wg(k)) # acquire measurements
   if Nopi <= Wg(k) then
      if gi>0 then
         let gi = gi + 1 #tune grain size
      else
         let Nopt = Nopi # tune num.workers
      end if
      exit for
   else
      let Tc(k) = (Tc(0)-slot) * (Tc(k)/Tc(0))
      let Nopi = calc(Nop in Tc(k))
      if Nopi < Wg(k) then
          let Nopt = Nopi # tune num.workers
         exit for
      else
         let Nw = Nw + Wg(k)
         let slot = slot + Wg(k) * (Tc(k)/Nopi))
      end if
   end if
end for
```

Number of Workers with Heterogeneous Processors

- Allocation of communication time slot using processor speed ratio.



Content

1. Motivation
2. Performance Model for Dynamic Tuning
3. Tuning Heuristics
4. Experimentation Results
5. Conclusions

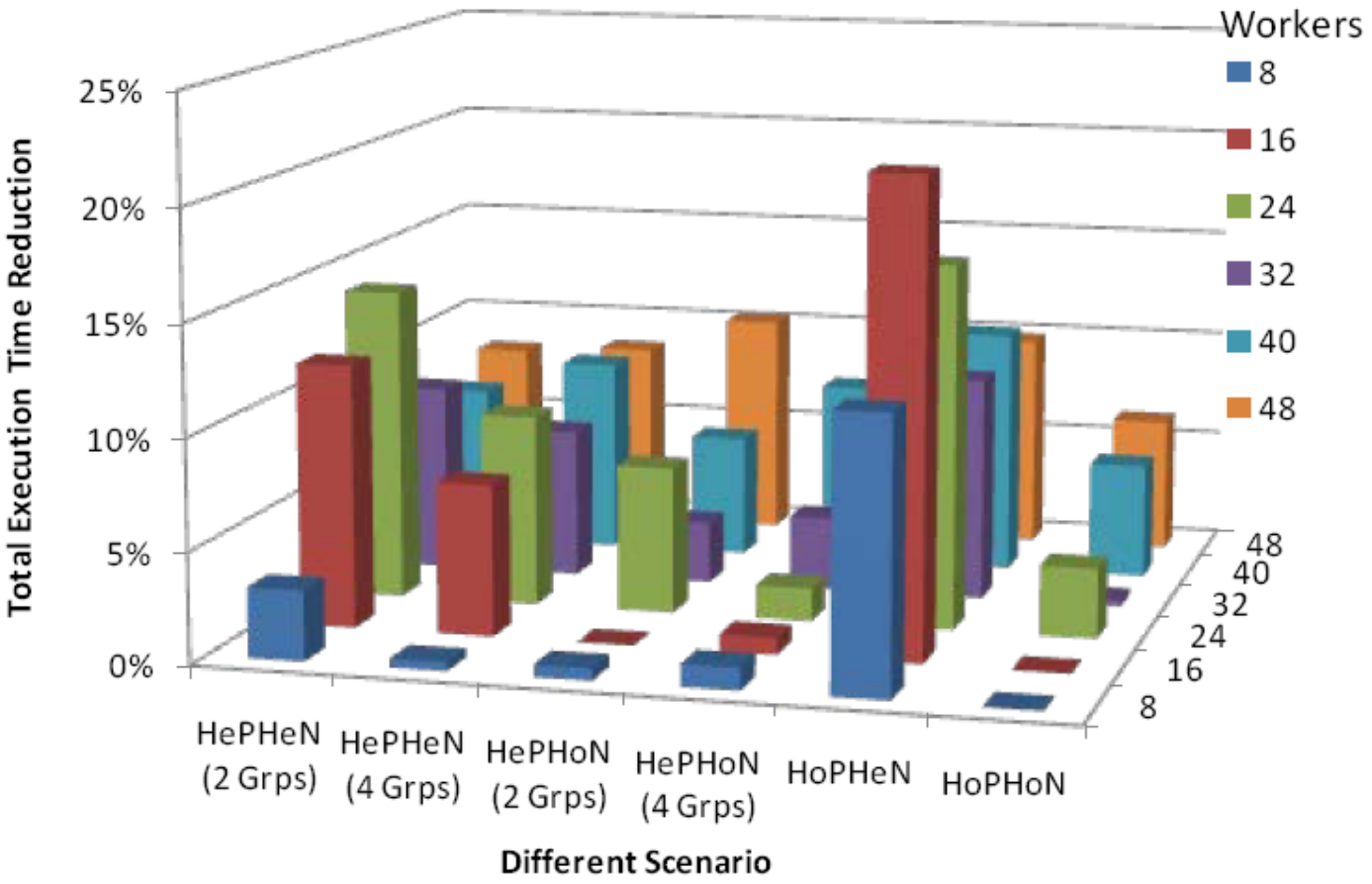
Experimentation Methodology

- Application execution may have different measurements in different executions.
- Cases coverage:
  - If grain size and number of worker tuning do not raise execution time in all different execution scenarios and system characteristics change, the new scenario found should also have benefits from the tuning.
- Event driven simulation using real task execution times and network bandwidth.

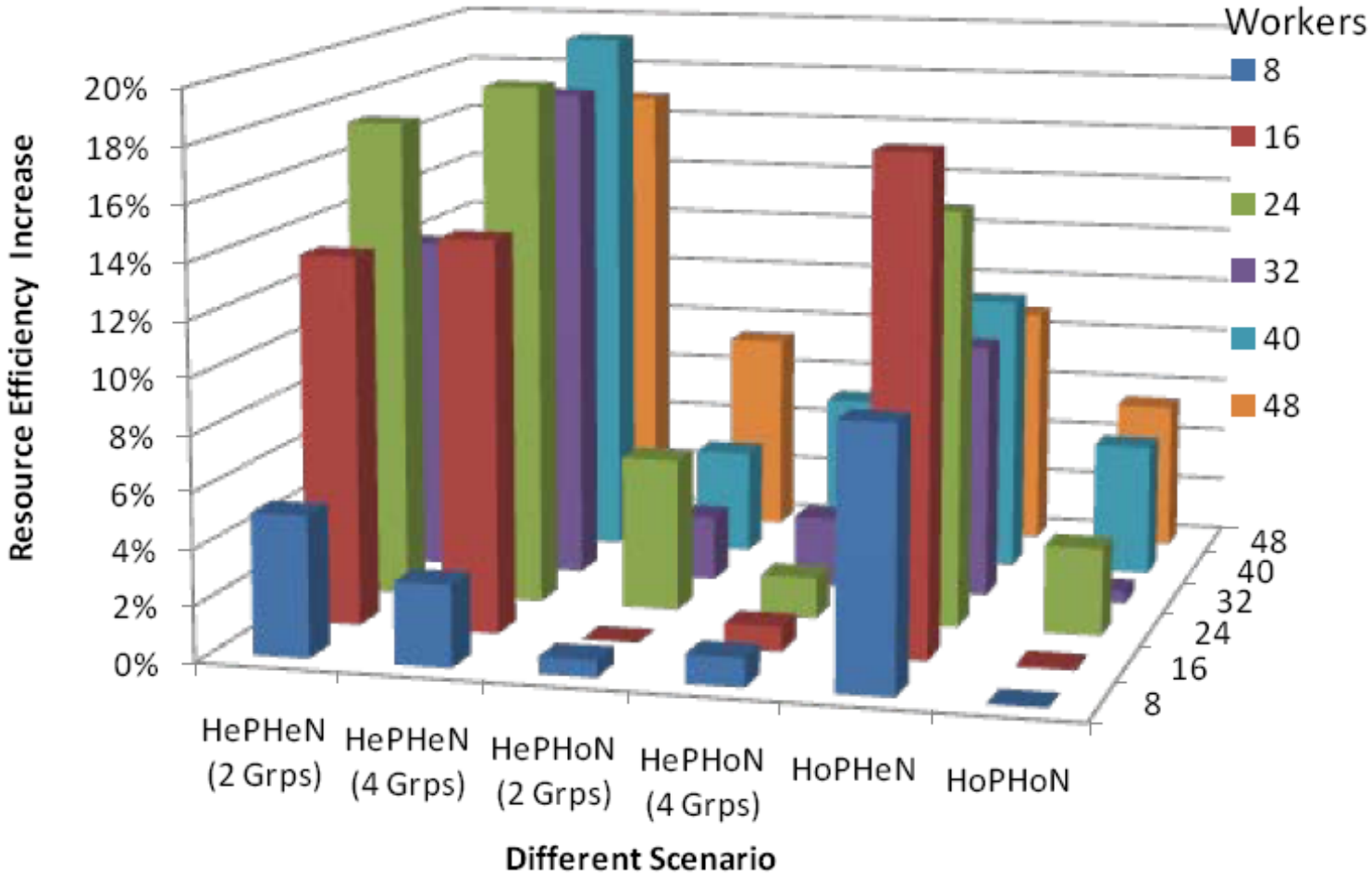
Workload

- Matrix Multiplication Application with block row and column division parallelization strategy with dynamic load balance (faster processors first).
- Simulation parameters
  - Task Execution time → ATLAS sgemm
  - Network Bandwidth
    - Local Workers→ MPI
    - Remote Workers → iperf tool
- Cases with processor and network heterogeneity and grain sizes from compute to communication bound scenarios.

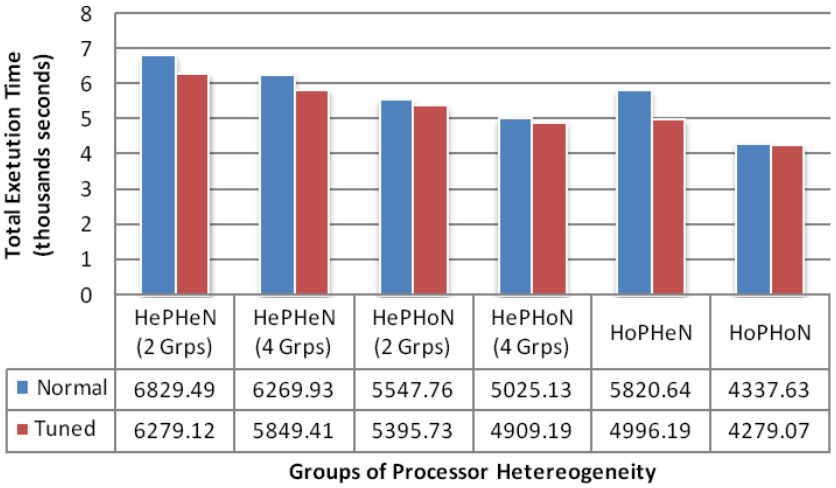
Total Execution Time Analysis



Resource Usage Efficiency Analysis

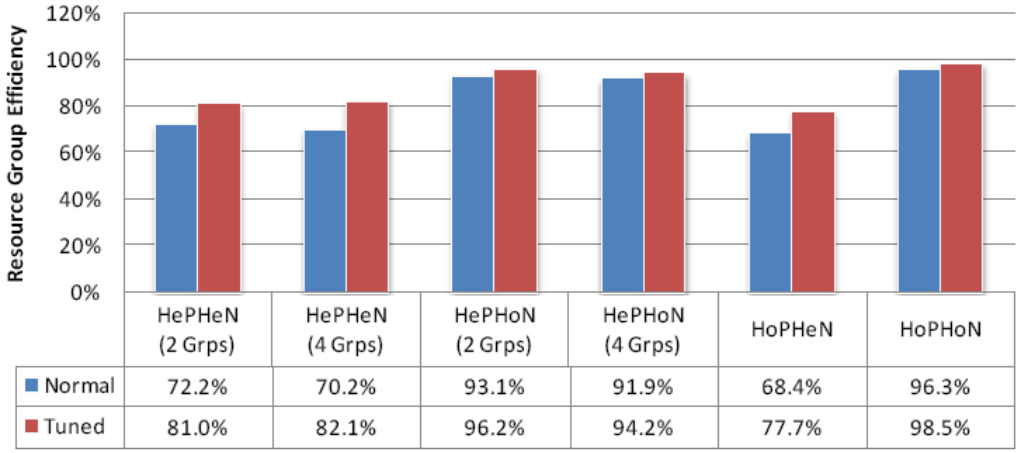


Heterogeneity Analysis



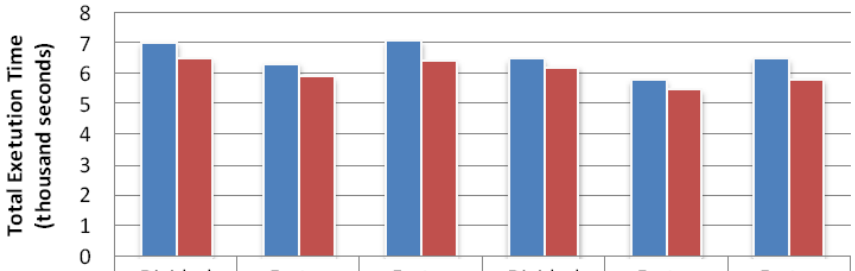
	HePHeN (2 Grps)	HePHeN (4 Grps)	HePHoN (2 Grps)	HePHoN (4 Grps)	HoPHeN	HoPHoN
Normal	6829.49	6269.93	5547.76	5025.13	5820.64	4337.63
Tuned	6279.12	5849.41	5395.73	4909.19	4996.19	4279.07

Groups of Processor Hetereogeneity



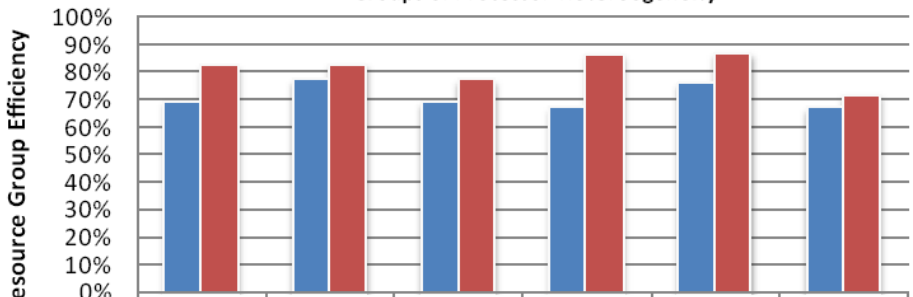
	HePHeN (2 Grps)	HePHeN (4 Grps)	HePHoN (2 Grps)	HePHoN (4 Grps)	HoPHeN	HoPHoN
Normal	72.2%	70.2%	93.1%	91.9%	68.4%	96.3%
Tuned	81.0%	82.1%	96.2%	94.2%	77.7%	98.5%

Groups of Processor Hetereogeneity



	Divided Equally (2 Grps)	Faster Local (2 Grps)	Faster Remote (2 Grps)	Divided Equally (4 Grps)	Faster Local (4 Grps)	Faster Remote (4 Grps)
Normal	7046.39	6360.86	7081.23	6501.05	5792.15	6516.58
Tuned	6498.53	5900.88	6437.96	6240.16	5488.83	5833.86

Groups of Processor Hetereogeneity LAN and WAN division



	Divided Equally (2 Grps)	Faster Local (2 Grps)	Faster Remote (2 Grps)	Divided Equally (4 Grps)	Faster Local (4 Grps)	Faster Remote (4 Grps)
Normal	69.9%	77.5%	69.3%	67.3%	76.2%	67.1%
Tuned	82.6%	83.2%	77.4%	86.8%	87.2%	72.1%

Groups of Processor Hetereogeneity LAN and WAN division

Content

1. Motivation
2. Performance Model for Dynamic Tuning
3. Tuning Heuristics
4. Experimentation Results
5. Conclusions

Conclusions

- We proposed a heuristic to dynamically change application grain size and number of workers.
  - Deal with network and processor heterogeneity.
  - Increase application performance indexes.
- The heuristic was evaluated over a Master/Worker application in different processor and network heterogeneous scenarios.
  - Computational Grids composed by commodity clusters scattered over Internet .
- The best gains where obtained on scenarios with higher level of heterogeneity.

Open Lines

- Apply our heuristic on different Grid master/worker applications.
- Use same heuristic to tune pipeline stage replication in pipeline execution over Grids.
- Grain Size selection on SPMD applications.
- Hierarchical dynamic tuning of grain size and number of workers.

Thank You!

Questions??