Adaptive resource remapping through live migration of virtual machines

Muhammad Atif

Peter Strazdins*

Research School of Computer Science

The Australian National University

Contents

Australian National University

- Introduction
- Related Work
- Resource Remapping Framework ARRIVE-F
  - Performance Model
  - Migration decisions
  - Implementation
- Experiments
- Conclusion

Introduction

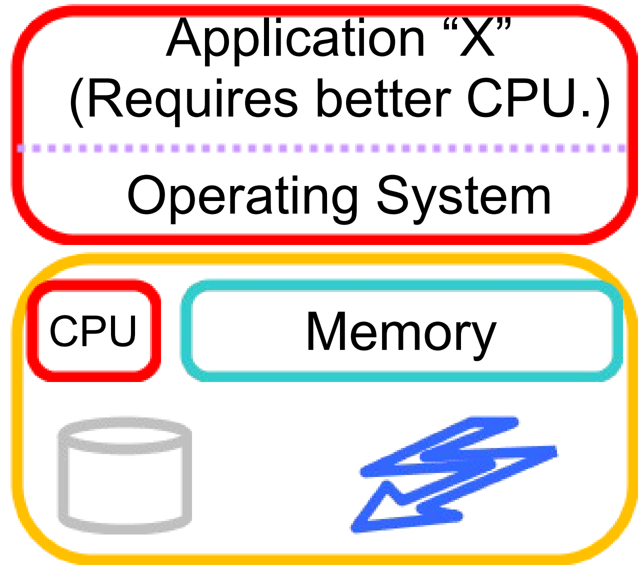
- Compute farms become heterogeneous.
  - Frequent upgrades, Specific nodes for projects.
  - CPU Speeds, Memory, Communication interfaces.
- Poor utilization.
  - Job requiring faster communication can land on nodes with slower interconnects.
- Effective mapping of jobs in such clusters is NP complete.
  - A number of heuristics proposed.

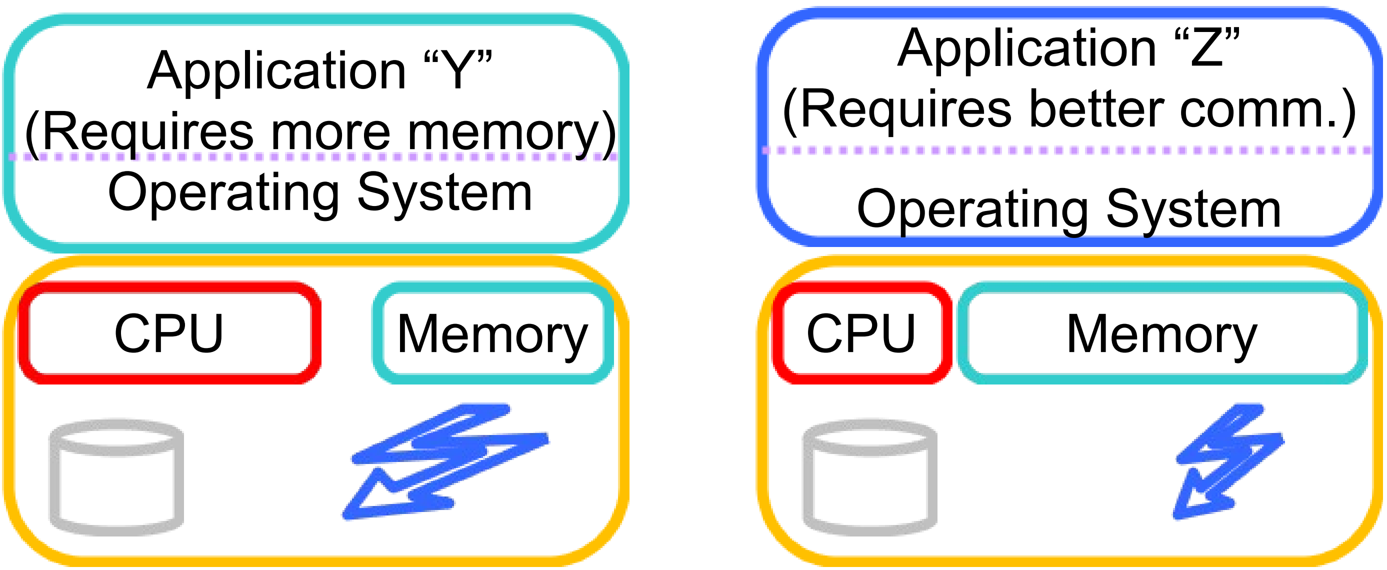
Pictorially

Profile for:

- CPU (Hardware Characteristics)
- Communication Characteristics
- Memory utilization.

Notice the unhappy systems

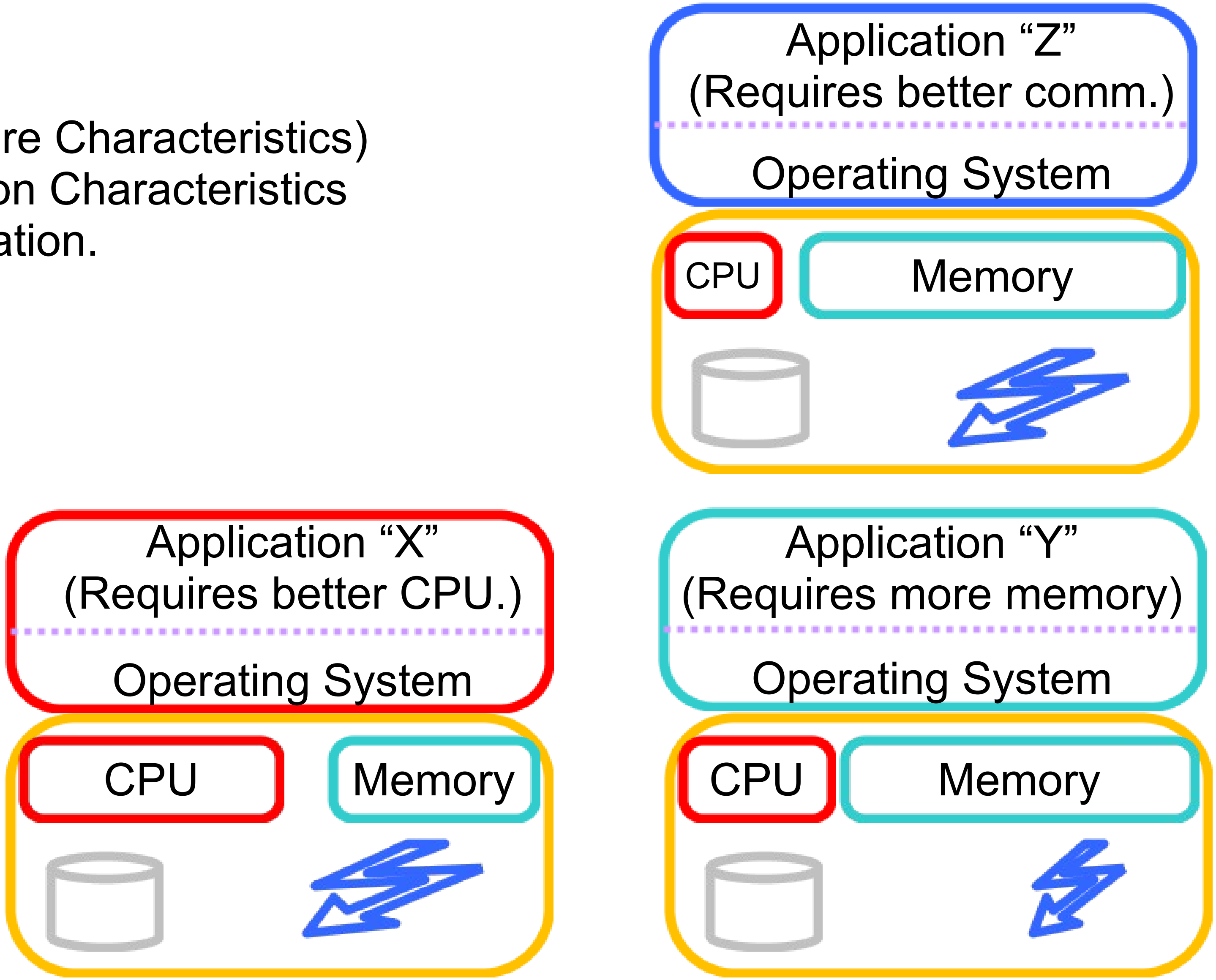




Pictorially

Profile for:

- CPU (Hardware Characteristics)
- Communication Characteristics
- Memory utilization.



Related Work

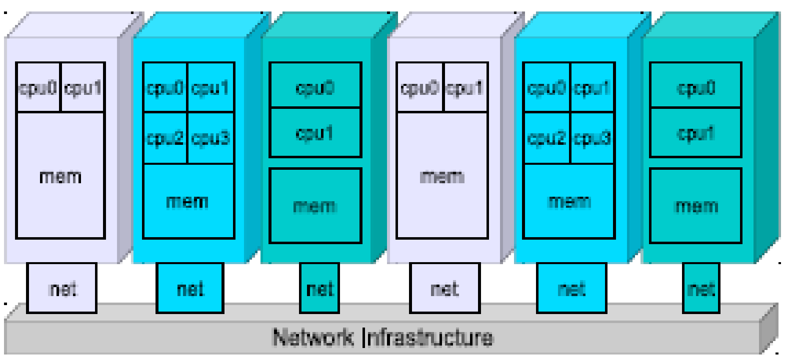
- Heterogeneity aware schedulers
  - Static cluster scheduling
  - Applications are scheduled based on their profile
  - Require off-line profiling
- Heterogeneity aware applications
  - Application distributes its load based on the cluster.
  - Source code modification
- Migration
  - Process migration using Mosix

Comparison with ARRIVE-F

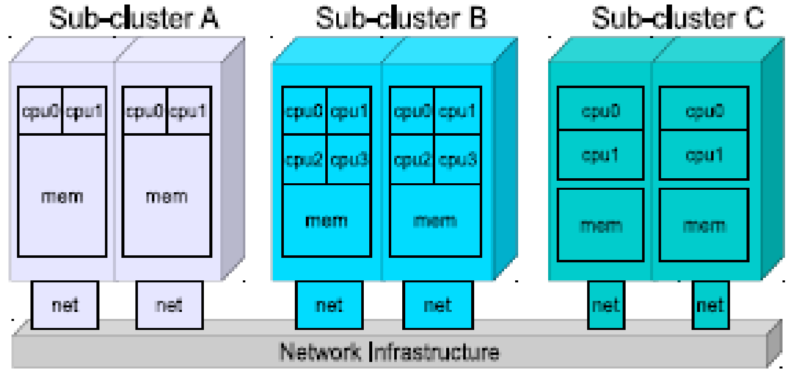
- ARRIVE-F does not require source code modifications
- No offline profiling mode
- Execution times based on real hardware metrics
  - L1/L2 Cache misses, Flops
- Live migration facility of hypervisor to migrate jobs to suitable clusters
- Can take advantage of dynamic conditions

Framework

- Assumptions
  - Iterative scientific applications
  - Run-times in the order of minutes
  - Do not cater for I/O intensive jobs
  - Heterogeneous compute farm divided into a number of homogeneous compute clusters.



(a) Heterogeneous compute cluster.



(b) Heterogenous compute farm with homogeneous sub-clusters.

Performance Modeling

- Online Performance Modeling
  - Computational Model
  - Communication Model
  - Memory utilisation Model
  - Migration Model

Computation Model

- Responsible for generating CPU profile of the running application
- Use L1/L2 and FLOPS; but not limited to these

$$t^{\mathrm{P}}_{A,j} = \sum_i Pctr_{A,j,i} \times \frac{Cycles_{A,i}}{f_A}$$

- Simple approximation

$$\tilde{t}^{\mathrm{P}}_{B,j} = \sum_i Pctr_{A,j,i} \times \frac{Cycles_{B,i,j}}{f_B}$$

Communication Model

- Two sub-models
  - Blocking and Non blocking
- Blocking Communication
  - Log the frequency of different message sizes
  - Multiplied by 'precomputed' latency of that message size

$$t^{\mathrm{B}}_{A,j} = \sum_{s} n^{\mathrm{B}}_{j}(s) \times l_{A}(s)$$

$$t^{\mathrm{B}}_{B,j} = \sum_{s} n^{\mathrm{B}}_{j}(s) \times l_{B}(s)$$

Communication Profile

- Non-blocking
  - Difficult; use a lightweight approximation
  - Record wait times by logging each MPI_Request with corresponding MPI_Wait

$$t^{\mathrm{N}}_{A,j} = \sum_{s} n^{\mathrm{N}}_{j}(s) \times w_A(s)$$

$$\tilde{t}^{\mathrm{N}}_{B,j} = \sum_{s} n^{\mathrm{N}}_{j}(s) \times w_A(s) \times \frac{l_B(s)}{l_A(s)}$$

Memory Utilization

- Swap thrashing is the most costly operation
- We migrate the application as soon as thrashing is detected.

Predicting Execution Time

Australian National University

- The time gained or lost by the job if it was executed on cluster ‘B’ can be obtained by subtracting the predicted computation and communication times for sub-cluster ‘B’ from the profiled times of sub-cluster ‘A’:

$$t_{A \rightarrow B,j} = (t^{\mathrm{P}}_{A,j} - \tilde{t}^{\mathrm{P}}_{B,j}) + (t^{\mathrm{B}}_{A,j} - t^{\mathrm{B}}_{B,j}) + (t^{\mathrm{N}}_{A,j} - \tilde{t}^{\mathrm{N}}_{B,j})$$

Migration Prediction

- Determine the time gained or lost w.r.t remaining time

$$T_{j,k}^{A\leftrightarrow B} = \eta_j t_{A\rightarrow B,j} \times \frac{T_j^{\text{rem}}}{\tau} + \eta_k t_{B\rightarrow A,k} \times \frac{T_k^{\text{rem}}}{\tau} - T_{j,k}^M$$

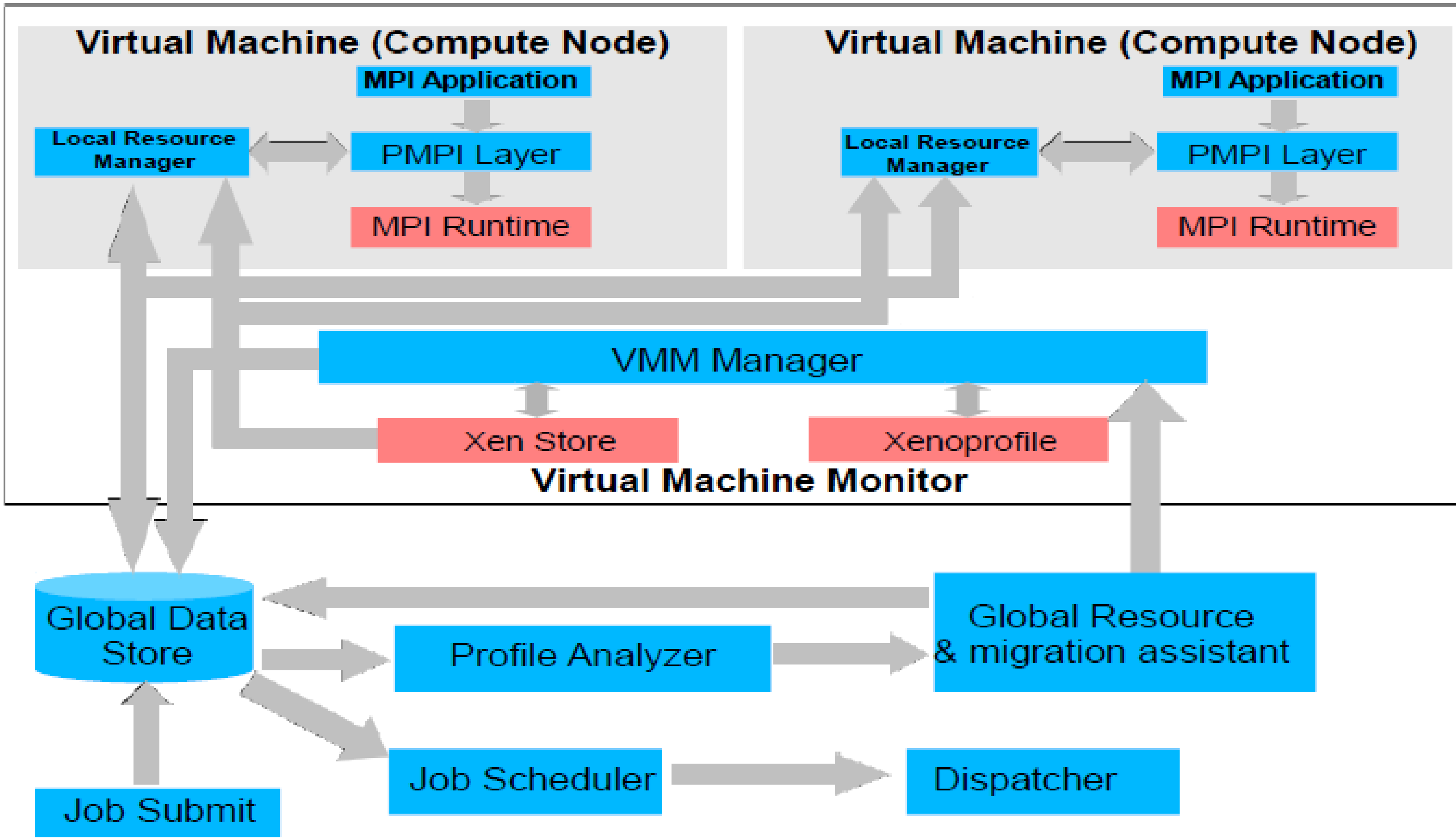
- Approximate w.r.t a time block

$$\bar{T}_{j,k}^{A\leftrightarrow B} = (\eta_j t_{A\rightarrow B,j} + \eta_k t_{B\rightarrow A,k}) \times \beta - T_{j,k}^M$$

- Migrate if the following threshold is met

$$\bar{T}_{j,k}^{A\leftrightarrow B} > T^{Thresh}$$

Implementation



Adaptive **R**esource **R**elocation **I**n **V**irtualized **E**nvironments – **F**ramework
Open source under GPL-v3 (http://cs.anu.edu.au/~muhammad.atif/opensource)

Experimental Evaluation

- Heterogeneous cluster
  - XEN 3.3 compiled from source;
    - XenoLinux 2.6.31.12
  - Live Migration Patch [5]
  - β=20 ; τ=50 ;

Cluster	CPU Type	Memory	**Total Machines**
A	4 × Opteron 2.2 Ghz	4 GB	2
B	4 × Phenom II 3.0 Ghz	4 GB	2
C	4 × Phenom II 3.0 Ghz	4 GB	2
D	2 × Athlon 2.0 Ghz	1.2 GB	4

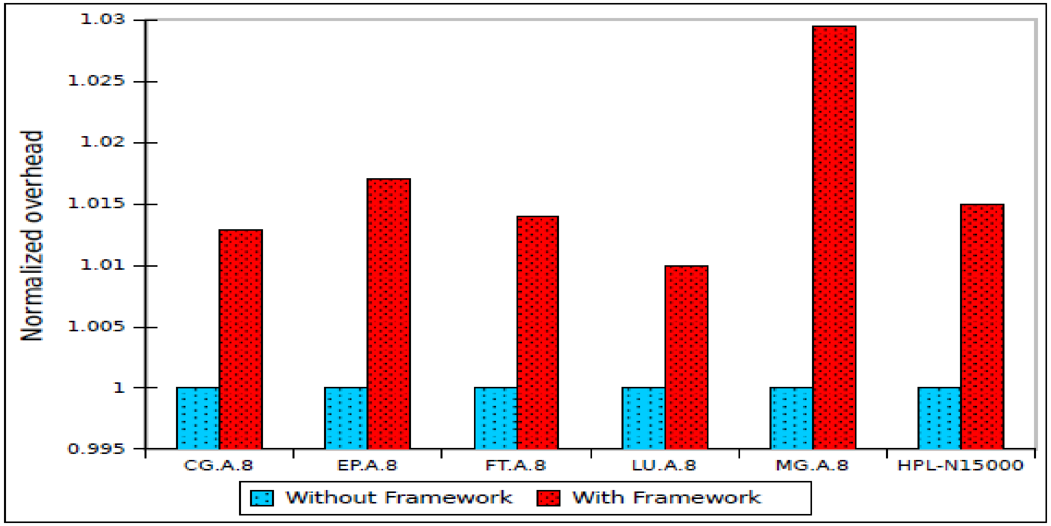
Accuracy and Overheads

Benchmark	T_A^{act}	T_B^{act}	$T_{A \to B}^{act}$	% Acc CPU	% Acc Prof
CG.B.8	104.5	57.9	71.2	75.5	81.3
FT.B.8	98.2	81.6	77.2	88.6	94.6
LU.B.8	240.7	81.3	103.4	46.0	78.6
HPL.N15K	150.7	62.2	68.5	56.2	90.8

Computational Accuracy

Benchmark	T_C^{act}	T_B^{act}	$T_{C \to B}^{act}$	% Acc Prof
CG.B.8	141.0	57.9	66.2	88.8
FT.B.8	375.1	81.6	79.3	97.2
LU.B.8	106.8	81.3	61.8	76.0
HPL.N15K	150.7	62.2	80.2	71.1

Communication Accuracy



Overheads of the framework

Throughput Experiments

- Lublin-Feitelson Method to generate workload
- NPB kernels CG, EP, FT, IS, LU and MG
  - Modified iterations to increase the wall-clock time
- Compare ARRIVE-F with FCFS-Backfill algorithm
  - Jobs allocated to fastest clusters if possible.
- A number of experiments conducted
  - Only one is being presented
- Each experiment was conducted 3 times
  - Averages presented.

Experiment 1

- Stream of 330 jobs
- Throughput improvement = 27%
- Time saved = 32%
- Average waiting time reduced by = 55%

Experiment 1

- Total of three migration decisions
  - Migration 1: Thrashing
  - Migration 2: Communication
  - Migration 3: CPU

Migration Number	Job Name	Sub-cluster	T^{est}	T^{act} Base	$T^{act}_{A \leftrightarrow B}$ Mig.
Migration 1	FT.B.4.20	D	92	1148 (D)	415
	FT.A.4.156	C	230	95 (C)	108
Migration 2	MG.B.8.5132	A	2697	2332 (A)	1769
	FT.B.8.506	B	2174	2226 (B)	2222
Migration 3	CG.B.4.2286	A	3268	3408 (D)	2043
	LU.A.1.7385	A	5869	5870 (A)	4161
	LU.A.8.12334	C	1850	1058 (B)	1838
	LU.B.1.455	A	1500	1850 (A)	1447

Further Experiments

- **Second experiment;**
  - FT.B.4.* removed; no thrashing
  - Total time saved = 7%
  - Average waiting time and turn around time = 1%
- **Third Experiment**
  - Removed cluster with FAST ethernet
  - Total time saved = 33%
  - Average waiting time improved = 298%
  - Turn around time improved = 230%

Conclusion

- Heterogeneity in compute farms can be successfully addressed by virtualization and migration (can easily extend to other classes of apps)
- Lightweight profiling
  - 3% overhead
- Applicable to Cloud Computing
- Green Computing
- Envision such online profiling and migration frameworks will become part of standard cloud deployment in future

QUESTIONS!