Statistical Performance provisioning and energy efficiency in distributed computing systems

Nikzad Babaii Rizvandi

Supervisors: Prof.Albert Y.Zomaya
Prof. Aruna Seneviratne

THE UNIVERSITY OF SYDNEY

NICTA

OUTLINE

- Introduction
- Background
- Our research direction
- Part 1 - Energy efficiency in Hardware using DVFS
  - Background
  - Multiple frequencies selection algorithm
  - Observations and mathematical proofs
- Part 2 - Statistical performance provisioning for jobs on MapReduce framework
  - Background
  - Statistical pattern matching for finding similarity between MapReduce applications → grouping/ classification
  - Statistical learning for performance analysis and provisioning of MapReduce jobs → modeling

INTRODUCTION

- World data centres used*
  - 1% of the total world electricity consumption in 2005
  - equivalent to about seventeen 1000MW power plants
  - growth of around 76% in five years period (2000-2005, 2005-2010)
- Improper usage of 44 million servers in the world **
  - Produce 11.8 million tons of carbon dioxide
  - Equivalent of 2.1 million cars

* Koomey, J.G. ,"Worldwide electricity used in data centres", Environ.Res. Lett. , 2008
** Alliance to Save Energy (ASE) and 1E, "Server Energy and Efficiency Report," 2009

INTRODUCTION

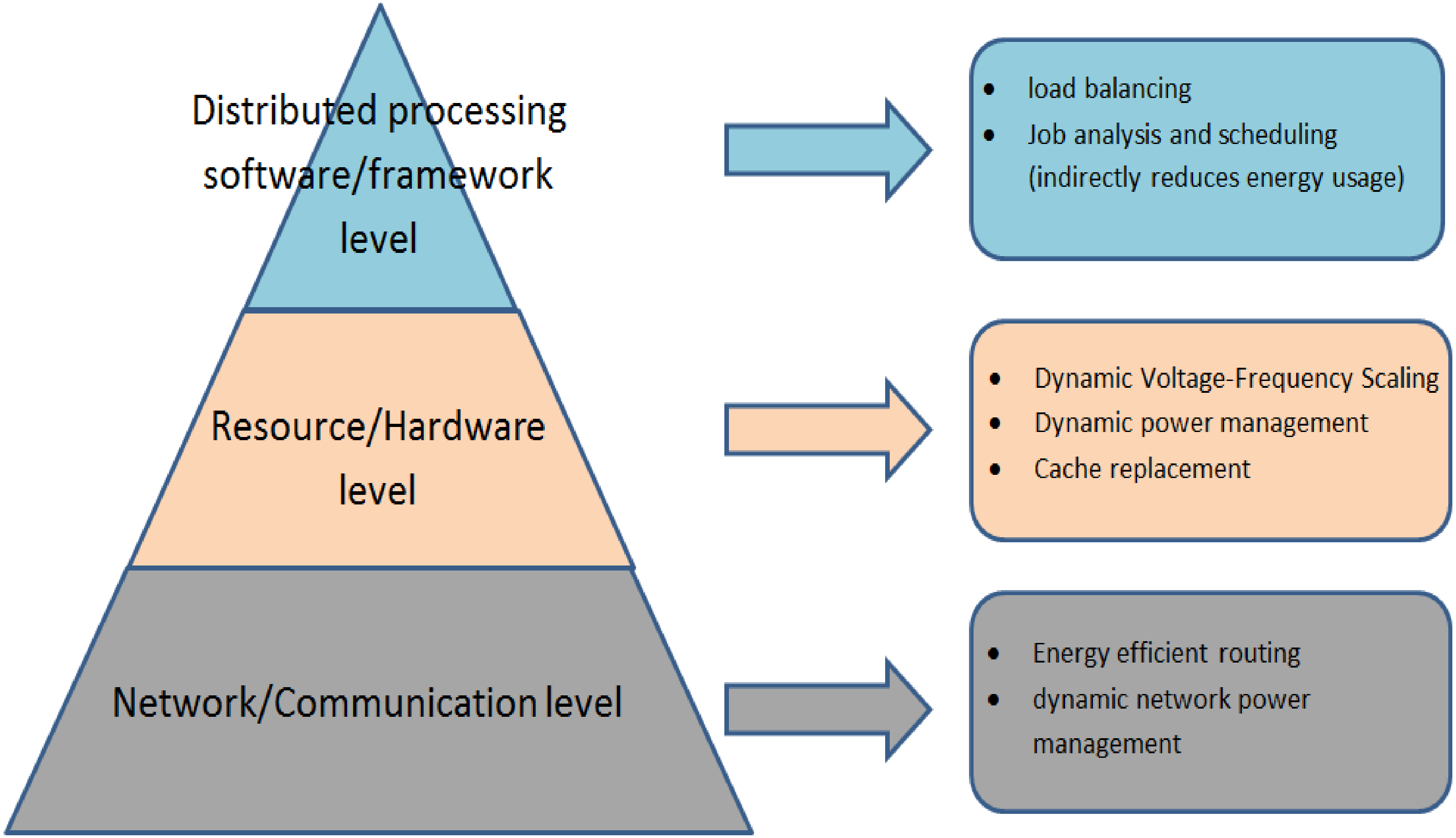
- U.S. data centres used 61 billion kWh of electricity in 2006**
  - 1.5% of all U.S. electricity consumption
  - double compare to 2000
  - Estimation of 12% growth per year

- Improper usage of server in U.S.
  - produces 3.17 million tons of carbon dioxide
  - Equal to 580,678 cars

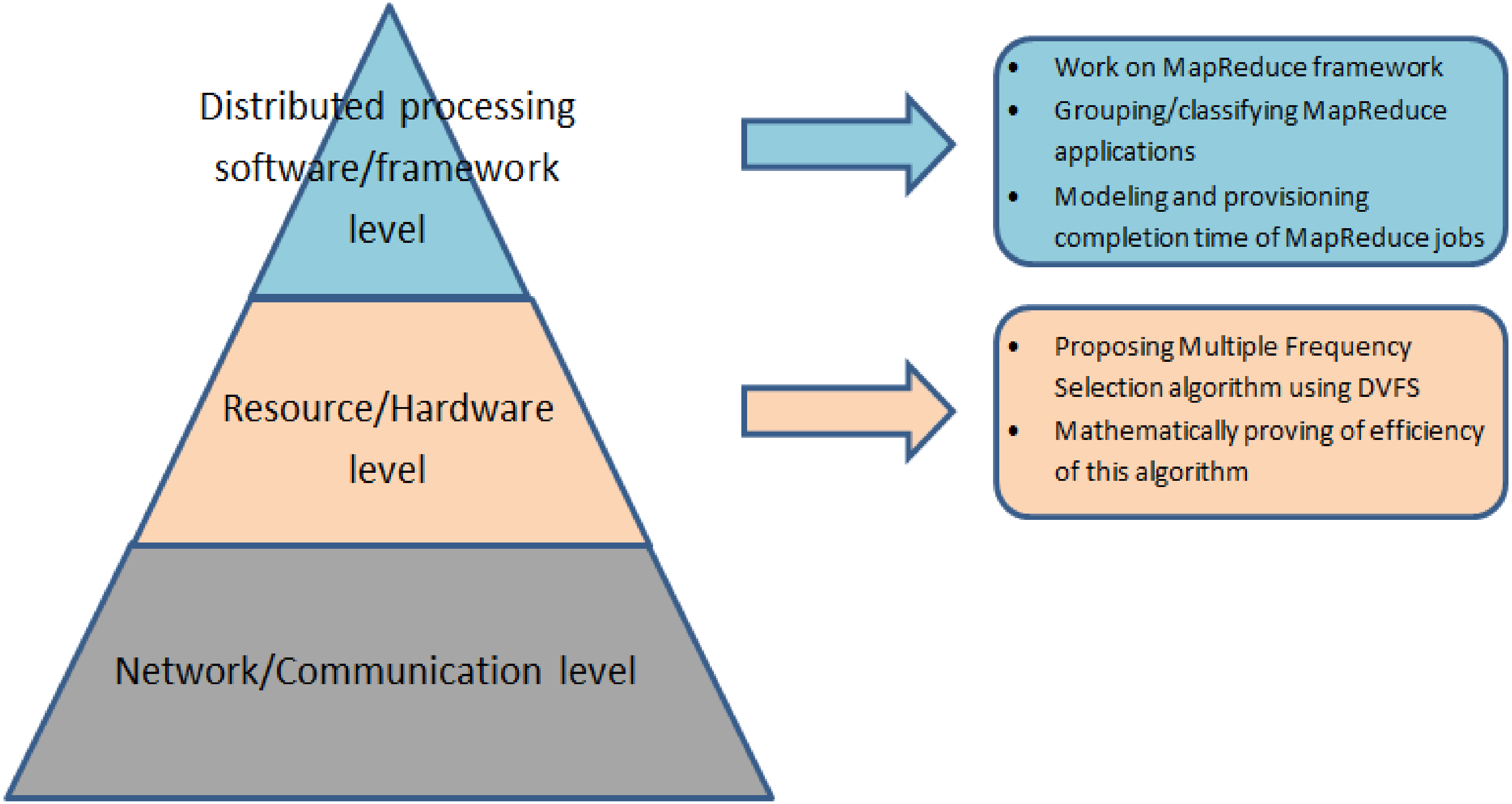
* Koomey, J.G. ,"Worldwide electricity used in data centres", Environ.Res. Lett. , 2008

** Alliance to Save Energy (ASE) and 1E, "Server Energy and Efficiency Report," 2009

ENERGY CONSUMPTION IN DISTRIBUTED SYSTEMS



OUR RESEARCH



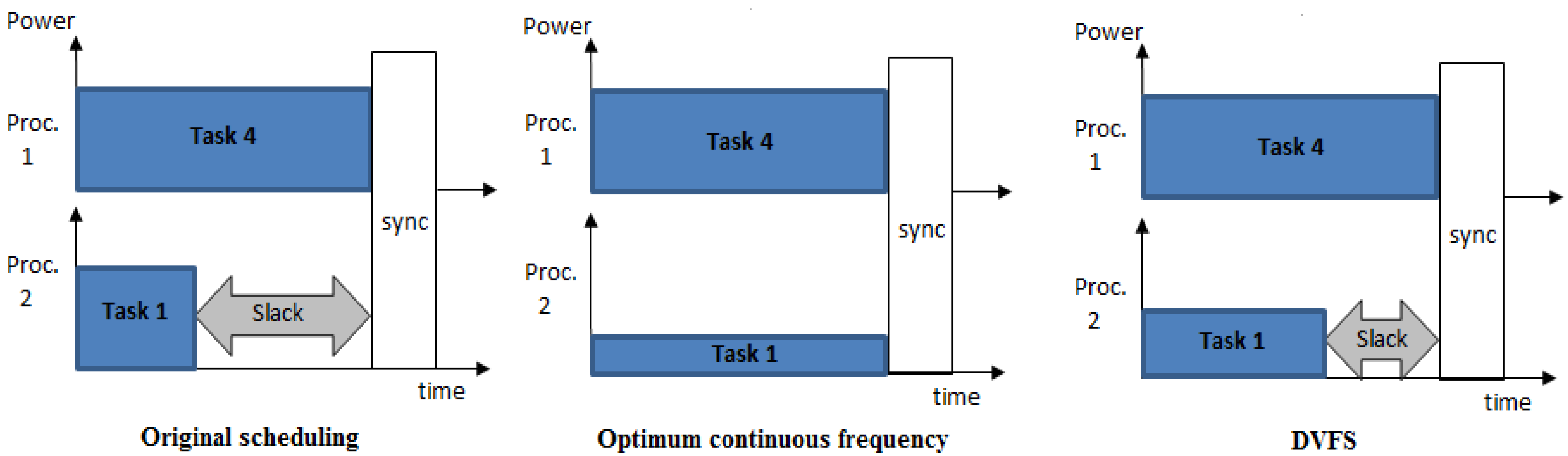
PART 1 - Energy Efficiency in Distributed Computing Systems Using DVFS

BACKGROUND – Dynamic Voltage Frequency Scaling

- In CMOS circuits: $\begin{cases} E = kfv^2t \\ f \propto v \end{cases} \Rightarrow E \propto f^3t$
- Example:

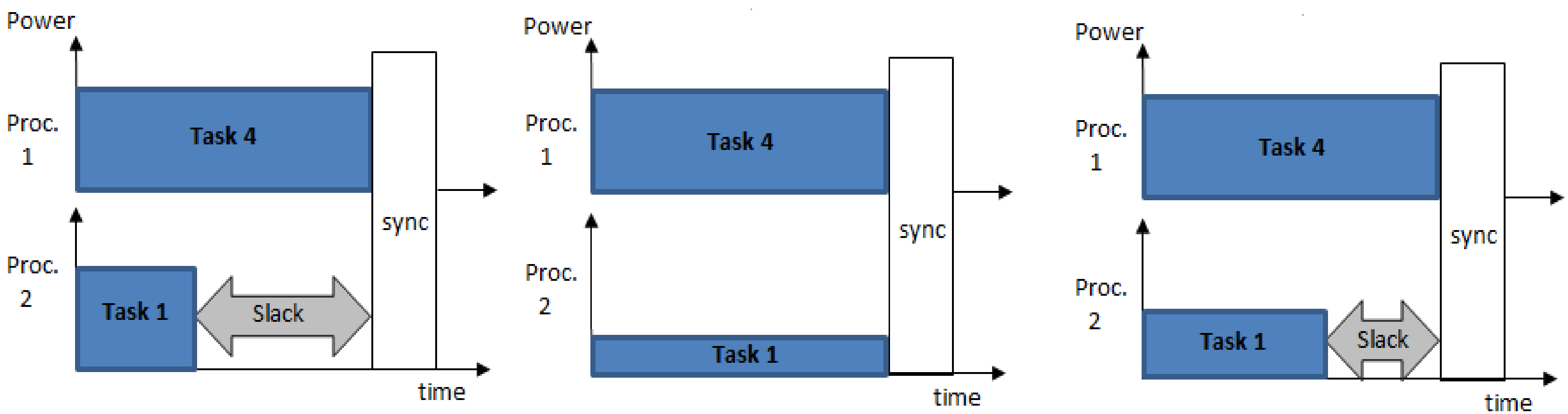
$$f \longrightarrow f' = \frac{f}{2} \Rightarrow \begin{cases} t' = 2t \\ E' = k(f')^2t' = k\left(\frac{f^3}{8}\right)(2t) = \frac{E}{4} \end{cases}$$

BACKGROUND – Dynamic Voltage Frequency Scaling



- Each task has a maximum time restriction.
- In hypothetical world
  - processors has continuous frequency.
  - For k^{th} task, Optimum Continuous Frequency ($f^{(k)}_{optimum}$) is a frequency that uses the maximum time restriction of the task.

BACKGROUND – Dynamic Voltage Frequency Scaling



Original scheduling **Optimum continuous frequency** **DVFS**

- In reality
  - processors has a discrete set of frequencies $\{f_1 > \cdots > f_N\}$.
  - The best frequency is slightly over $f_{optimum}^{(k)}$.

BACKGROUND – Energy-Aware Task Scheduling Using DVFS

- Research question:

  What is the suitable frequency selection to schedule tasks on a set of processors (1) to meet tasks' time restrictions, (2) to consume less energy

- Energy-aware scheduling scheme:
  - *Optimize a new cost function including energy*

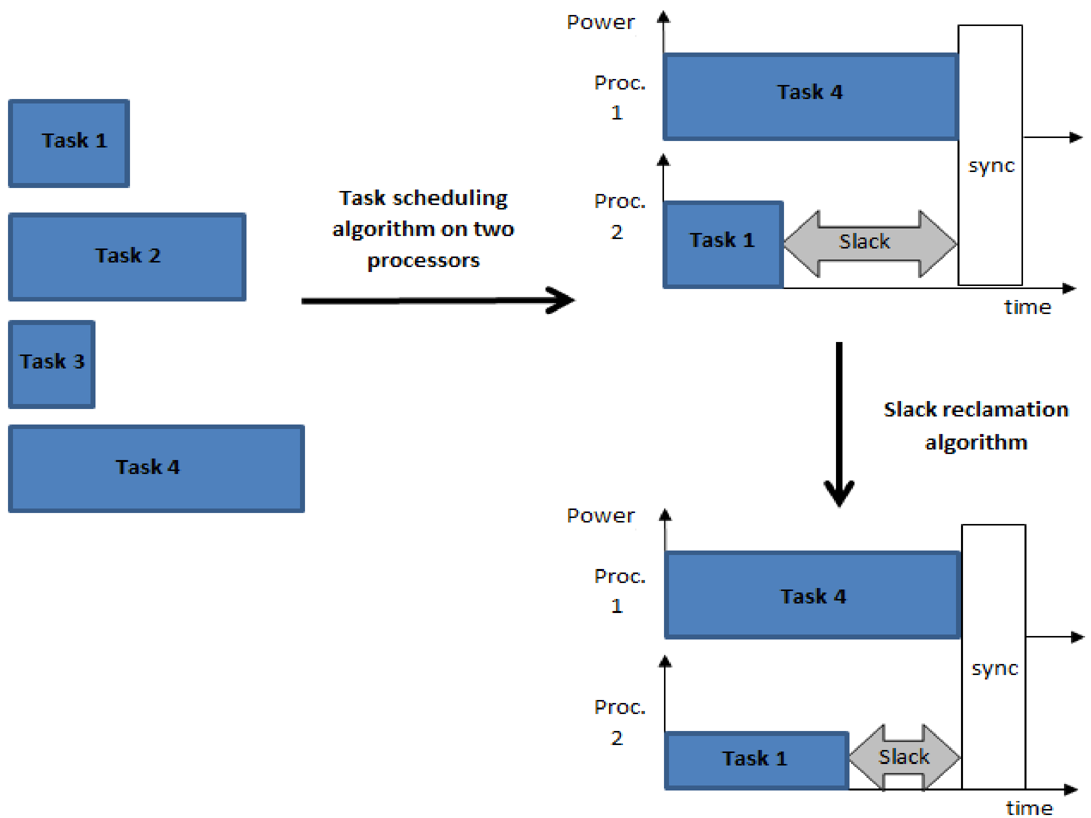
$$Cost\ function = f(Makespan, energy)$$

e.g.,

$$Cost\ function = \alpha \times Makespan + \beta \times Energy$$

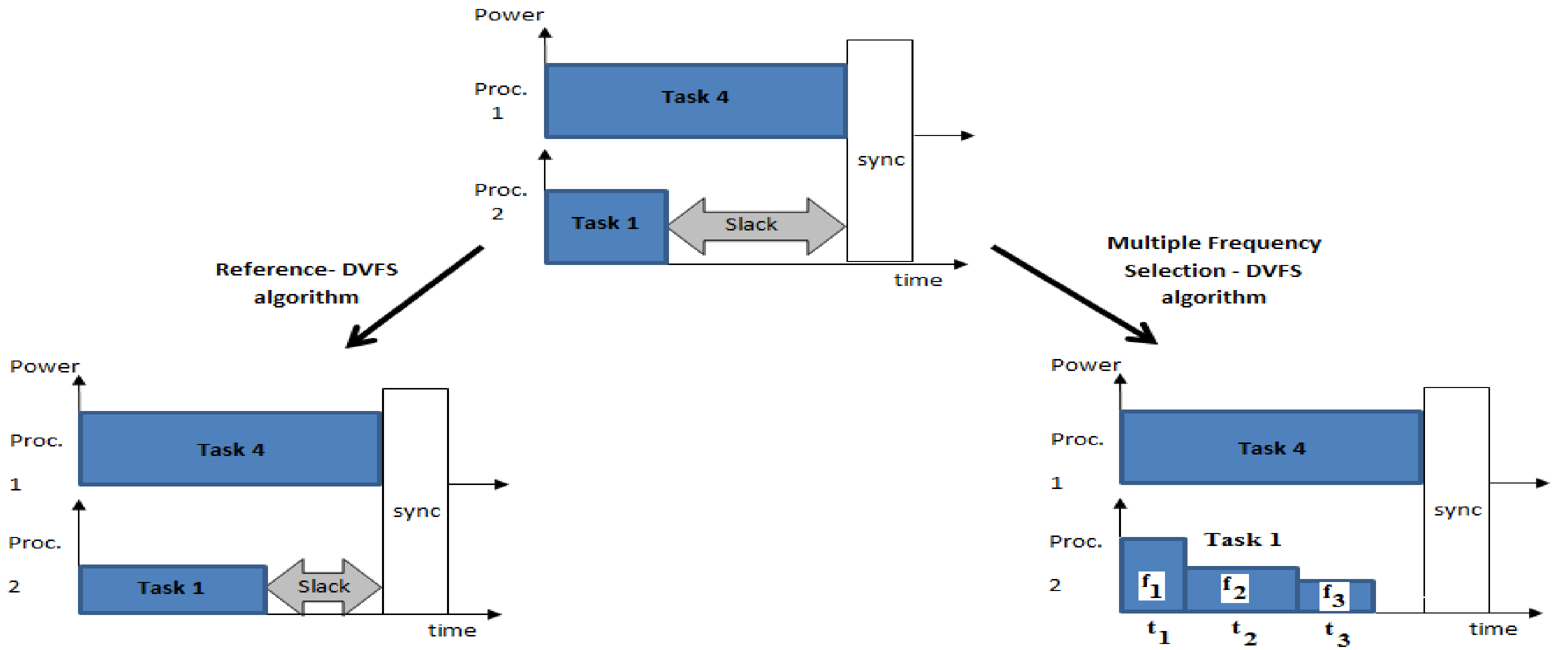
BACKGROUND – Energy-Aware Task Scheduling Using DVFS

- Slack reclamation
  - On top of scheduled tasks
  - Any slack on processors is used to reduce the speed of running task



OUR WORK – Multiple Frequency Selection (MFS-DVFS) for Slack Reclamation

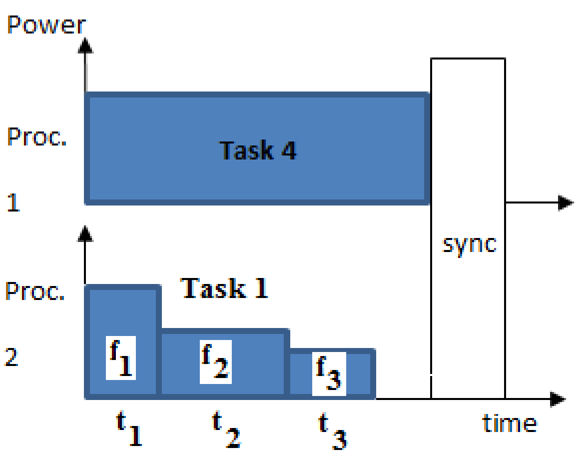
- Current scheme: use one frequency for a task (RDVFS algorithm)
- Our idea: use linear combination of all processor's frequencies for each task.



OUR WORK - Optimization Problem in MFS-DVFS

- In literature: $P_d(f,v) = kfv^2$
- For k^{th} task in scheduling

$$\begin{cases} Minimize : E^{(k)} = \sum_{i=1}^{N} t_i^{(k)} P_d(f_i, v_i) + P_{Idle}\left(T^{(k)} - \sum_{i=1}^{N} t_i^{(k)}\right) \\ \quad subject\ to: \\ \qquad 1. \sum_{i=1}^{N} t_i^{(k)} f_i = K^{(k)} \\ \qquad 2. \sum_{i=1}^{N} t_i^{(k)} \leq T^{(k)} \\ \qquad 3. t_i^{(k)} \geq 0, \quad for\ \ i = 1,2,\ldots,N \end{cases}$$



MFS-DVFS – Algorithm

Input: *the scheduled tasks on a set of P processors*

1. ***For*** k^{th} *task* ($A^{(k)}$) *scheduled on processor* P_j
2. *Solve optimization problem by linear programming*
3. ***end for***
4. **return** *(the voltages and frequencies of optimal execution of* the task)

OBSERVATION – Mathematical Proofs*

- In literature: $P_d(f,v) = kfv^2$
- Generalization: $P_d(f,v)$
  - If $(f_i, v_i) < (f_j, v_j)$ then $P_d(f_i, v_i) < P_d(f_j, v_j)$
- Observation #1
  - In hypothetical world, the cont. frequency that uses maximum time restriction of k^{th} task gives the optimum energy saving ($f^{(k)}_{optimum}$)
- Observation #2
  - For k^{th} task, always up to two voltage-frequencies are involved in optimal energy consumption ($f_i < f_j$)
  - Proof: theorem 1, 2

* N.B.Rizvandi, et al., "Some observations on optimal frequency selection in DVFS-based energy consumption minimization", J. Parallel Distrib. Comput. 71(8): 1154-1164 (2011)

OBSERVATION – Mathematical Proofs*

- Observation #3
  - $f_i < f_{optimum}^{(k)} < f_j$
  - Proof: lemma 1, 2
- Observation #4
  - The associated time for these frequencies (t_i, t_j) is

$$\begin{cases} t_i^{(k)} = \frac{K^{(k)} - T^{(k)} f_i}{f_j - f_i} \\ t_j^{(k)} = \frac{T^{(k)} f_j - K^{(k)}}{f_j - f_i} \end{cases}$$

  - Proof: Corollary 1

* N.B.Rizvandi, et al., "Some observations on optimal frequency selection in DVFS-based energy consumption minimization", J. Parallel Distrib. Comput. 71(8): 1154-1164 (2011)

OBSERVATION – Mathematical Proofs*

- Observation #5
  - The consumed energy of processor for this task associated with these two voltage-frequencies is

$$E^{(k)} = \frac{T^{(k)} f_j - K^{(k)}}{f_j - f_i} P_d(f_i, v_i) + \frac{K^{(k)} - T^{(k)} f_i}{f_j - f_i} P_d(f_j, v_j)$$

  - Proof: Corollary 2
- Observation #6
  - Using less time to execute the task results in more energy consumption
  - Proof: theorem 3

* N.B.Rizvandi, et al., "Some observations on optimal frequency selection in DVFS-based energy consumption minimization", J. Parallel Distrib. Comput. 71(8): 1154-1164 (2011)

OBSERVATION – Mathematical Proofs*

- In a simplified version, $f \propto v$ then
$$P_d(f, v) = \lambda f^3$$
- Observation #7
  - These two frequencies are neighbors. i.e., two immediate frequencies around $f_{optimum}^{(k)}$
  - Proof: theorem 4, 5

* N.B.Rizvandi, et al., "Some observations on optimal frequency selection in DVFS-based energy consumption minimization", J. Parallel Distrib. Comput. 71(8): 1154-1164 (2011)

MFS-DVFS – New Algorithm

Input: *the scheduled tasks on a set of P processors*

1. For *k^{th} task $(A^{(k)})$ scheduled on processor P_j*

2. *Calculate $f_{optimum}^{(k)}$*

3. Select the neighbour frequencies in the processor's frequency set before and after $f_{optimum}^{(k)}$. These frequencies are $f_{RD}^{(k)}$ and $f_{RD-1}^{(k)}$.

4. Calculate associated times and energy consumption.

5. Select $\left(f_{RD}^{(k)}, f_{RD-1}^{(k)}\right)$ associated to the lowest energy for this task

6. end for

return (individual frequencies pair for execution of each task)

EXPERIMENTAL RESULTS - Simulator

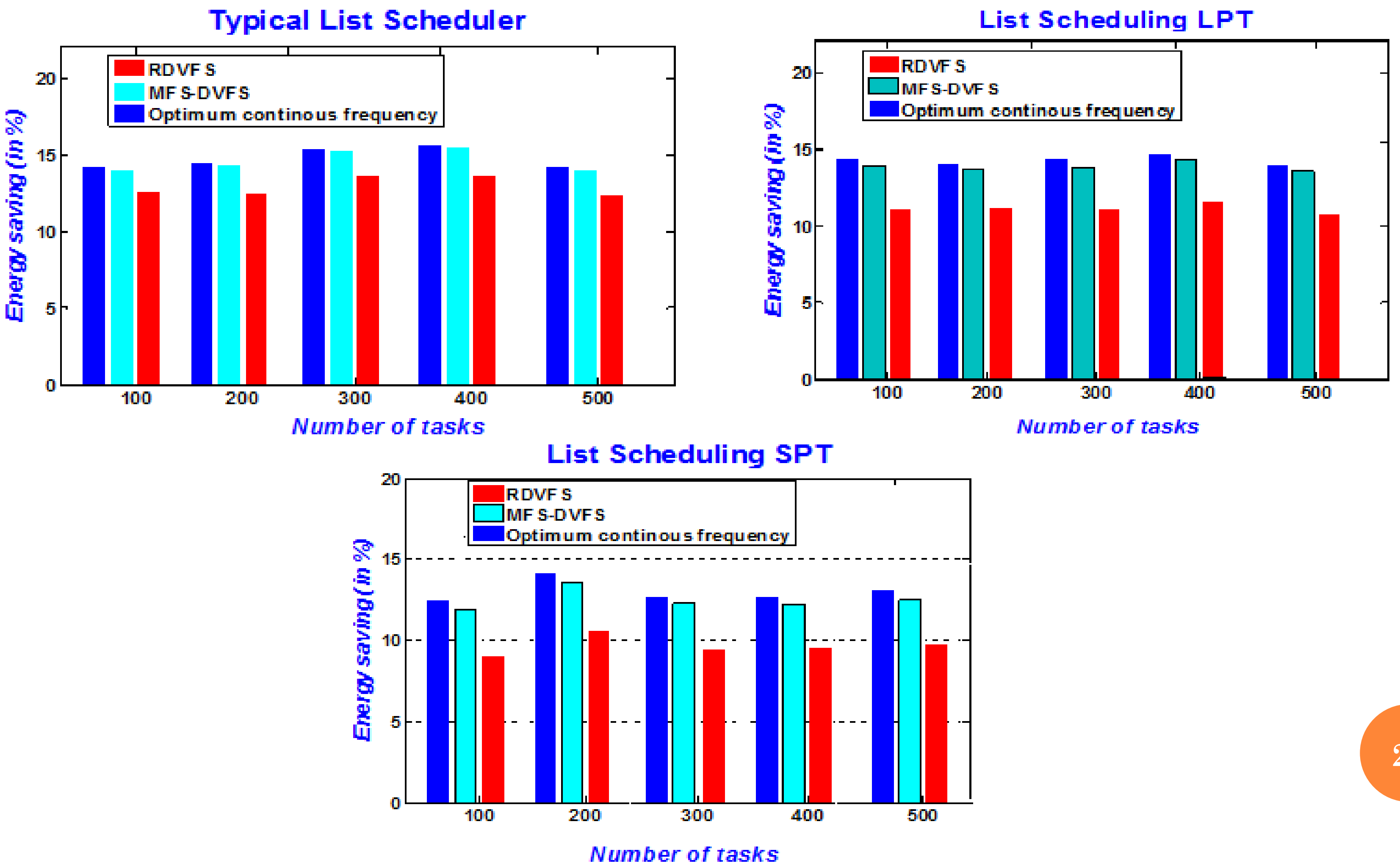
- Simulation settings
  - Scheduler
    - List scheduler
    - List scheduler *with Longest Processing Time First (LPT)*
    - List scheduler *with Shortest Processing Time First (SPT)*
  - Processor power model *
    - Transmeta Crusoe
    - Intel Xscale
    - Two synthetic processors

EXPERIMENTAL RESULTS - Simulator

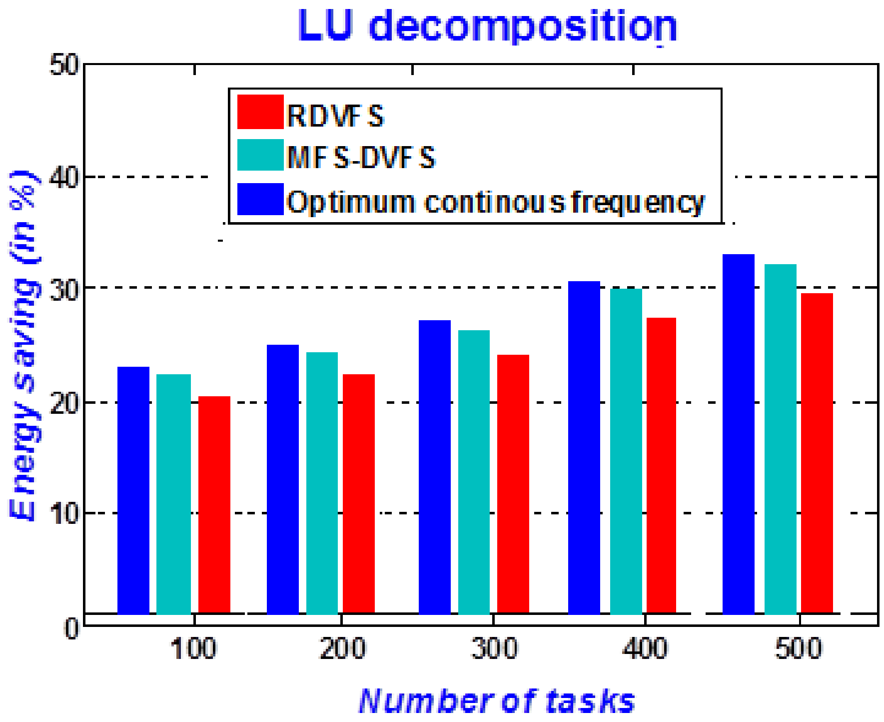
- Task graphs (DAG)
  - Random
  - LU-decomposition
  - Gauss-Jordan
  - Assumption: switching time between frequencies can be ignored compare to task execution time
- Experimental parameters

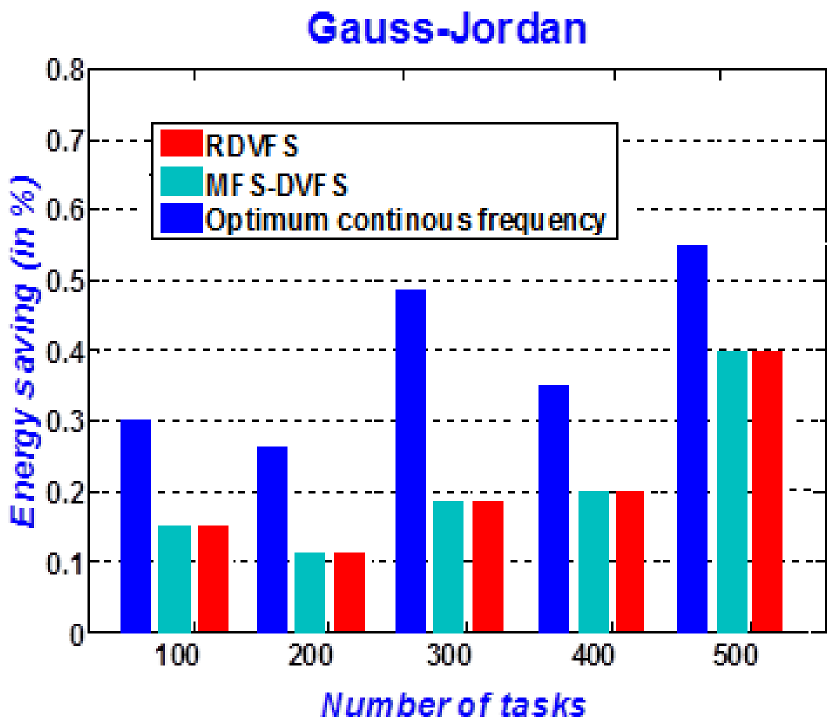
Parameter	Value
# of tasks	*100, 200, 300, 400, 500*
# of processors	*2, 4, 8, 16, 32*

EXPERIMENTAL RESULTS – Results (1)



EXPERIMENTAL RESULTS – Results (2)





EXPERIMENTAL RESULTS – Results (3)

Experiment	Random tasks	Gauss-Jordan	LU-decomposition
RDVFS	13.00%	0.1%	24.8%
MFS-DVFS	14.40%	0.11%	27.0%
Optimum Continuous Frequency	14.84%	0.14%	27.81%

PART 2 – Performance analysis and provisioning in MapReduce clusters

BACKGROUND – MapReduce

- A parametric distributed processing framework for processing large data sets.
- Introduced by Google in 2004.
- Typically used for distributed computing on clusters of computers.
- Widely used in Google, Yahoo, Facebook and LinkedIn.
- Hadoop is famous open source version of MapReduce developed by Apache

MOTIVATION

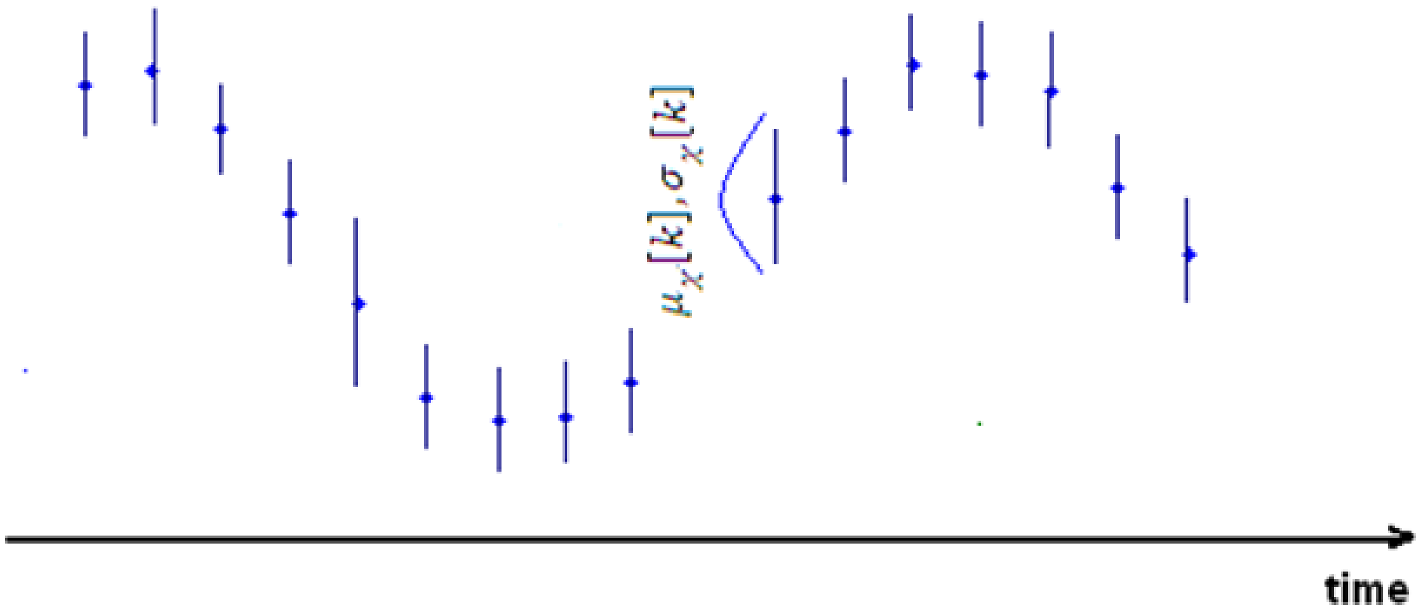
- Many users have job *completion time goals*
- There is strong correlation between completion time and values of configuration parameters
  - No support from current service providers, e.g. Amazon Elastic MapReduce
  - Sole responsibility of user to set values for these parameters.
- Our research work
  - Calculate similarity between MapReduce applications. It is too likely similar applications show similar performance for the same values of configuration parameters.
  - Estimate a function to model the dependency between completion time and configuration parameters by using Machine Learning techniques on historical data.

CLASSIFICATION

- Classification
  - Two MapReduce applications belong to the same group if they have *similar* CPU utilization pattern for several identical jobs.
  - An identical job in two applications means they run with
    - the same values for configuration parameters
    - the same size of small input data.
- hypothesis
  - Similar Applications share the same optimal values for the configuration parameters.
    - obtain the optimal values of configuration parameters for one application and use for others in the same group.

CLASSIFICATION - Uncertainty

- Uncertainty in CPU utilization pattern
  - Variation in values of each point in CPU utilization pattern for identical jobs of an application



CLASSIFICATION – Similarity (1)

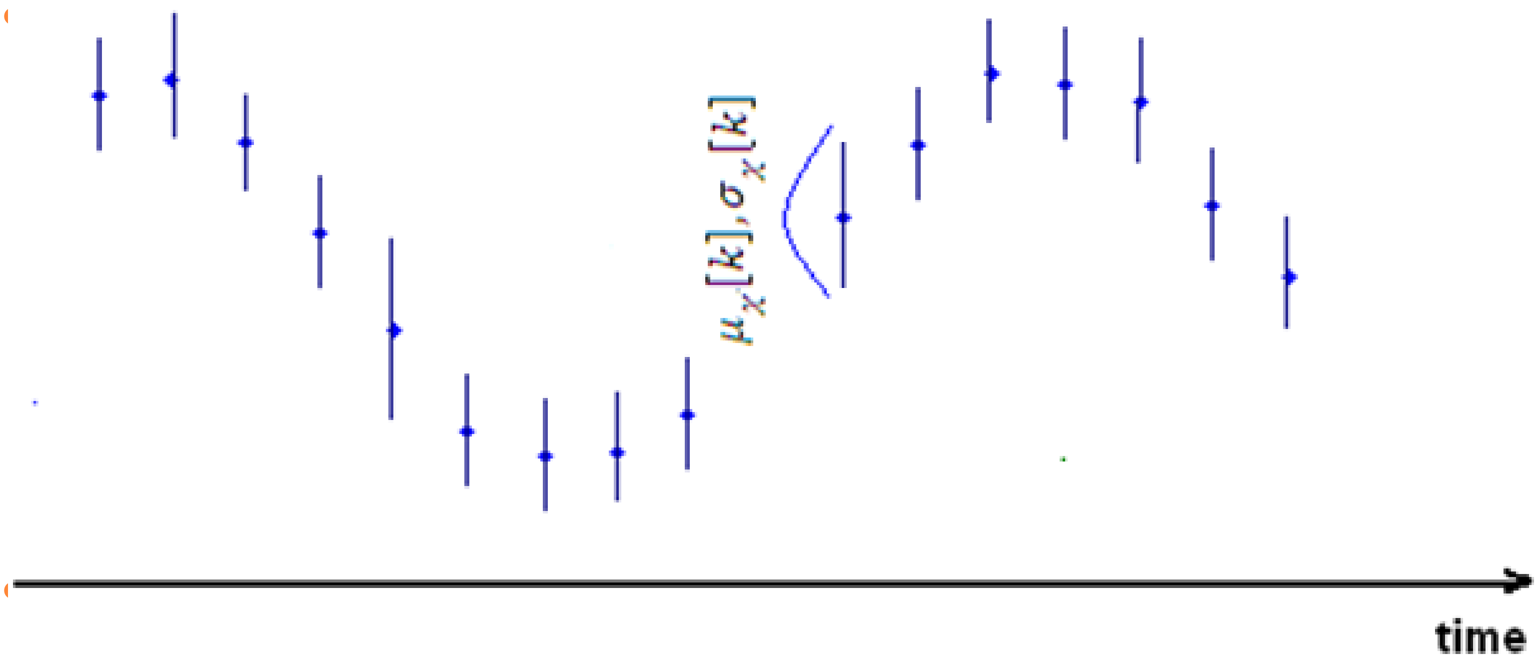
- High similarity is equal to low distance
- Current scheme
  - Average values of each point in patterns
  - Apply Dynamic Time Warping (DTW) on average CPU patterns → patterns become the same length
  - Calculate Euclidean distance between two average patterns (i.e., $\varphi_{avr.}$ and $\chi_{avr.}$)

$$\sum_{i=1}^{N}(\varphi_{avr.}[i] - \chi_{avr.}[i])^2 \leq r$$

  - r predefined distance threshold

CLASSIFICATION – Similarity (2)

- Our idea*
  - Comes from computational finance background.
  - Each point in the pattern has a Gaussian distribution.



* N.B.Rizvandi, et al., "A Study on Using Uncertain Time Series Matching Algorithms in Map-Reduce Applications", Concurrency and Computation: Practice and Experience, 2012

CLASSIFICATION – Similarity (3)

- Calculate $r_{boundry}$ as

$$r_{boundry} = \frac{r_{boundry,norm}^2 - \sum_{i=1}^{N} E(D^2[i])}{\sqrt{\sum_{i=1}^{N} Var(D^2[i])}}$$

$$r_{boundry,norm} = \sqrt{2} \times \mathrm{erf}^{-1}(2\tau - 1)$$

- If choose r $\leq r_{boundry}$, this guarantees that

$$\mathrm{P}\big(DST(\varphi_u, \chi_u) \leq r_{boundry}\big) \geq \tau$$

- So, $r_{boundry}$ is the minimum distance between two patterns with probability τ

$$\mathrm{P}\big(DST(\varphi_u, \chi_u) = r_{boundry}\big) = \tau$$

* N.B.Rizvandi, et al., "A Study on Using Uncertain Time Series Matching Algorithms in Map-Reduce Applications", Concurrency and Computation: Practice and Experience, 2012

CLASSIFICATION – technique

Set pre-defined Probability threshold $(\tau = 0.95)$

For the same input data size and configuration parameters' values

Run applications φ and χ ten times

Calculate mean and variance of φ and χ at each point in pattern

Endfor

Calculate joint mean and variance of distance between φ and χ from Eqns. 5 and 6*

Calculate $r_{boundry}$ from Eqn.10*

Two applications with lowest $r_{boundry}$ has highest similarity and can be in the same group.

* N.B.Rizvandi, et al., "A Study on Using Uncertain Time Series Matching Algorithms in Map-Reduce Applications", Concurrency and Computation: Practice and Experience, 2012

EXPERIMENTAL RESULTS – Setting

- Hadoop cluster settings
  - Five servers, dual-core
  - Xen Cloud Platform (XCP)
  - Xen-API to measure performance statistics
  - 10 Virtual machines
- Application settings
  - Four legacy applications: WordCount, TeraSort, Exim Mainlog parsing, Distributed Grep
  - Input data size
    - 5GB, 10GB, 15GB, 20GB
  - # of map/reduce tasks
    - 4, 8, 12, 16, 20, 24, 28, 32
  - Total number of runs in our experiments
    - $8 \times 8 \times 5 \times 4 \times 10 \times 4 = 51200$

EXPERIMENTAL RESULTS – Results (1)

Exim MainLog parsing

WordCount		*S*-1	*S*-2	*S*-3	*S*-4
	S-1	**24044**	117017	94472	228071
	S-2	80648	***64063***	58351	138222
	S-3	79431	**63232**	**54309**	104255
	S-4	147014	83655	81434	**70427**

Exim MainLog parsing

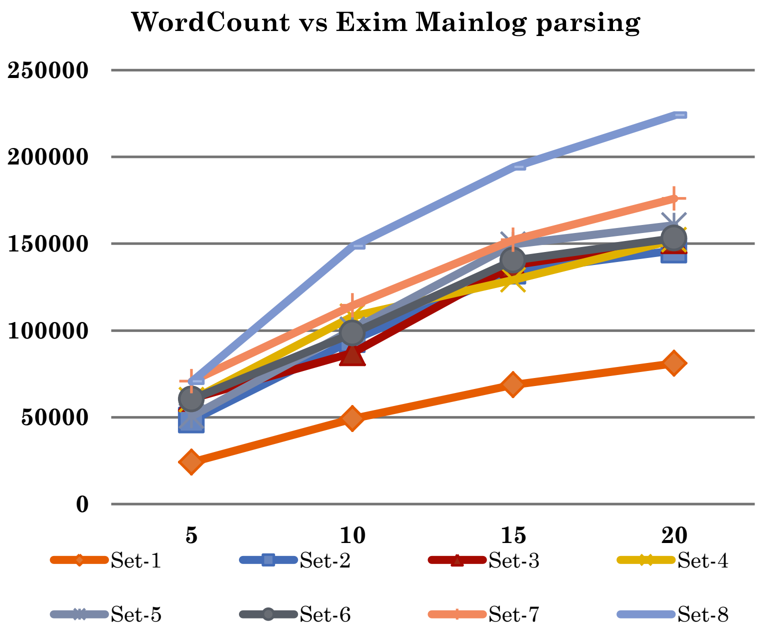
Terasort		*S*-1	*S*-2	*S*-3	*S*-4
	S-1	**27400**	***65102***	65606	132799
	S-2	155038	***67293***	68455	**70198**
	S-3	123668	76859	**56114**	76589
	S-4	166234	77829	81751	***74693***

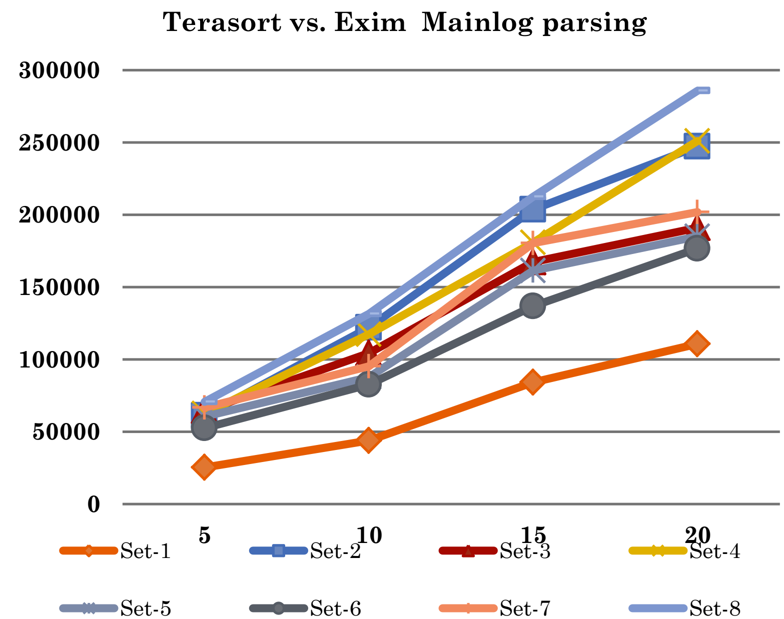
WordCount

Distributed Grep		*S*-1	*S*-2	*S*-3	*S*-4
	S-1	**21529**	105309	90012	199451
	S-2	79965	**62890**	68553	122279
	S-3	77549	62949	**51876**	101280
	S-4	142703	83089	72987	**69927**

$r_{boundry}$ between the applications for $\tau = 0.95$ for 5G of input data on 10 virtual nodes.

EXPERIMENTAL RESULTS – Results (2)





The size of input data vs. $r_{boundry}$ for $\tau = 0.95$ for 5G, 10G, 15G and 20G of input data on 10 virtual nodes.

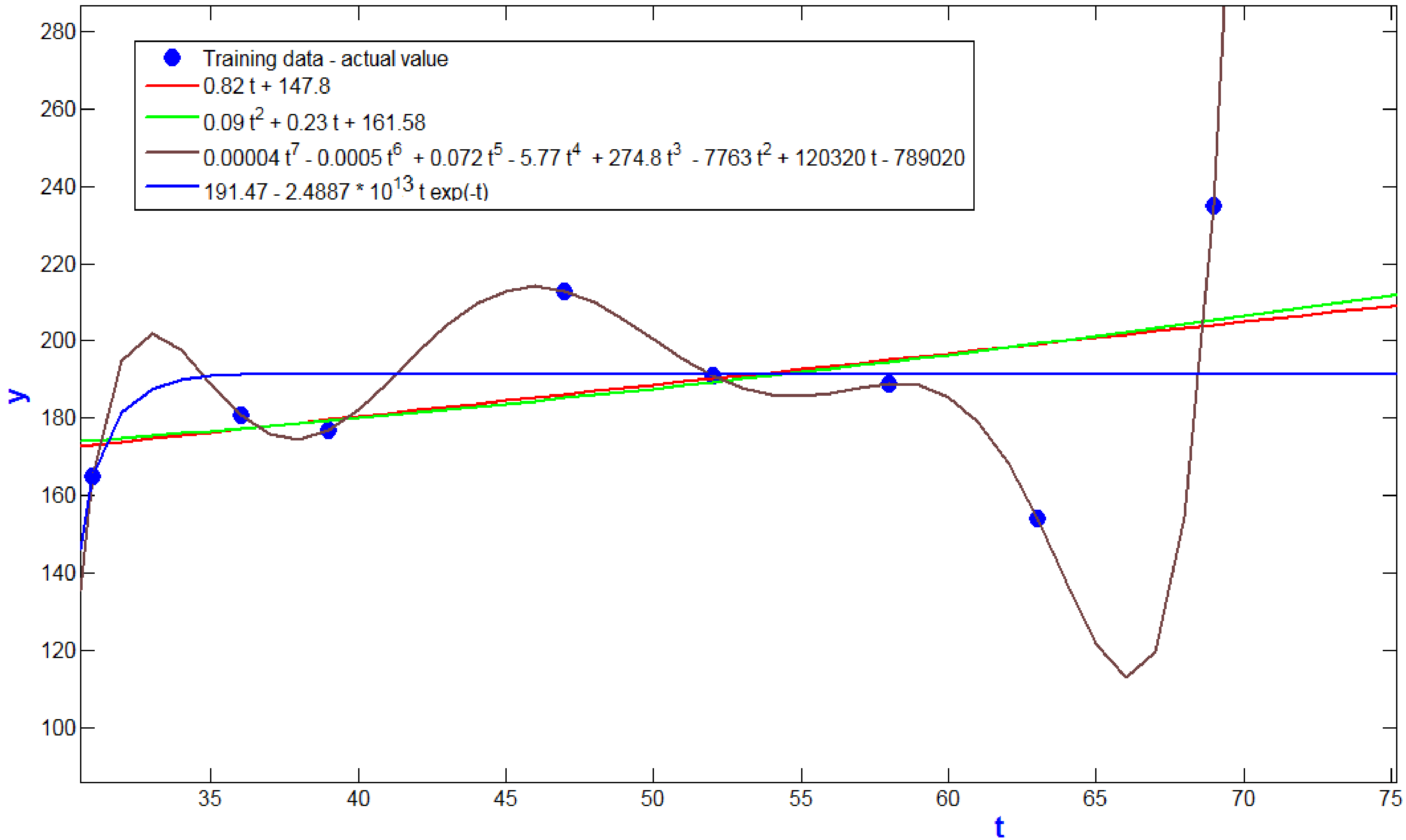
HOW TO USE THIS TECHNIQUE

- For frequent running jobs (e.g., Indexing, Sorting, Searching), run jobs with different values of conf. parameters
- For a new job, try to find most similar application in DB (using this algorithm). Then use its optimal values of parameters for running the new job.

MOTIVATION

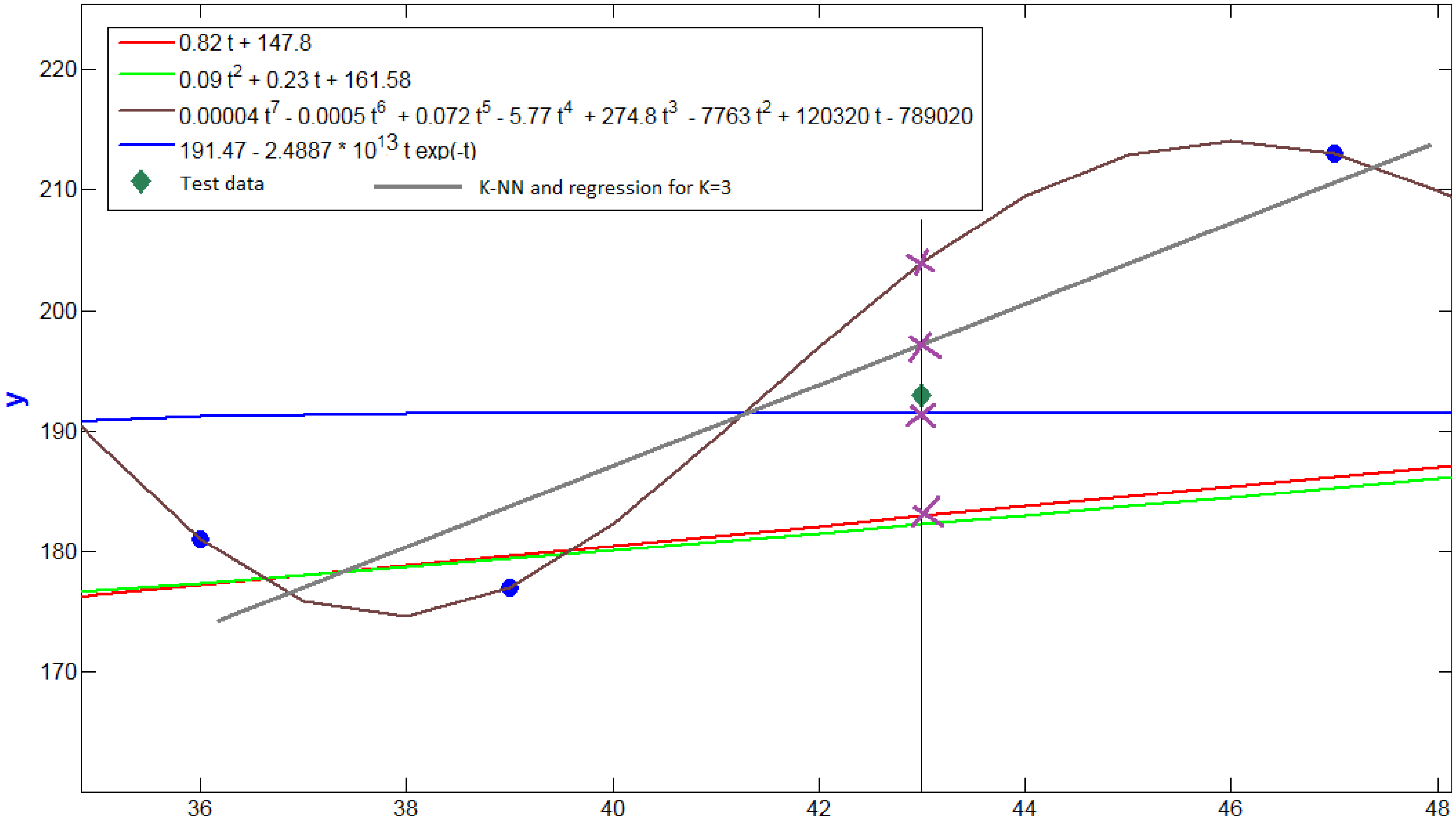
- Many users have job *completion time goals*
- There is strong correlation between completion time and values of configuration parameters
  - No support from current service providers, e.g. Amazon Elastic MapReduce
  - Sole responsibility of user to set values for these parameters.
- Our research work
  - Calculate similarity between MapReduce applications. It is too likely similar applications show similar performance for the same values of configuration parameters.
  - Estimate a function to model the dependency between completion time and configuration parameters by using Machine Learning techniques on historical data.

PERFORMANCE MODELLING AND PROVISIONING – Our idea (1)



Cholesterol level (y-axis) vs. Age (x-axis)

PERFORMANCE MODELLING AND PROVISIONING – Our idea (2)



Cholesterol level (y-axis) vs. Age (x-axis)

PERFORMANCE MODELLING AND PROVISIONING – Our idea (3)

- Prediction accuracy
  - Mean Absolute Percentage Error (MAPE)

$$MAPE = \frac{\sum_{i=1}^{M} \frac{\left| t_{exc.}{}^{(i)} - \widehat{t_{exc.}}{}^{(i)} \right|}{t_{exc.}{}^{(i)}}}{M}$$

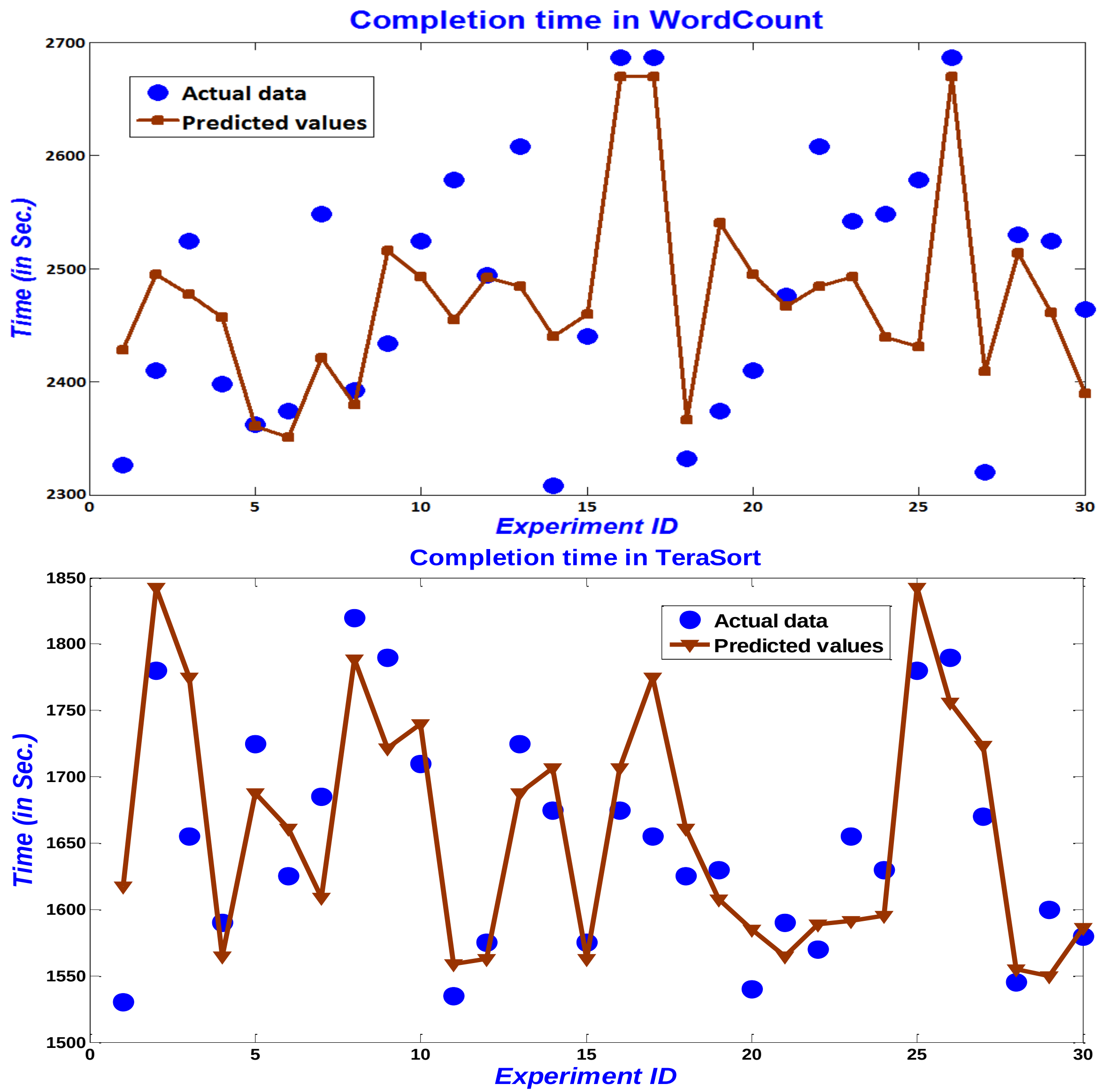
  - R^2 prediction accuracy

$$R^2 = 1 - \frac{\sum_{i=1}^{M} \left(t_{exc.}{}^{(i)} - \widehat{t_{exc.}{}^{(i)}}\right)^2}{\sum_{i=1}^{M} \left(t_{exc.}{}^{(i)} - \sum_{r=1}^{M} \frac{t_{exc.}{}^{(r)}}{M}\right)^2}$$

EXPERIMENTAL RESULTS – Setting

- Hadoop cluster settings
  - Five servers, dual-core
  - SysStat to measure performance statistics
  - Hadoop 0.20.0
- Application settings
  - Three applications: WordCount, TeraSort, Exim Mainlog
  - Input data size
    - 3GB, 6GB, 8GB, 10GB
  - # of map/reduce tasks
    - 80 number randomly chosen between 4 to 100
    - 56 experiments for model estimation, rest for model testing

EXPERIMENTAL RESULTS– Results (1)



EXPERIMENTAL RESULTS– Results (3)

	MAPE	R^2 prediction accuracy
WordCount	1.51%	0.83
Exim MainLog parsing	3.1%	0.76
TeraSort	2.33%	0.79

HOW TO USE THESE TECHNIQUES

- For frequent running jobs (e.g., Indexing, Sorting, Searching), run jobs with different values of conf. parameters
- Fit a model (based on second technique) and calculate the optimal values of parameters which minimizes completion time.
- Put this application + optimal values into DB
- For a new job, try to find most similar application in DB (using first technique). Then use its optimal values of parameters for running the new job.
- Reducing completion time indirectly reduces energy consumption.

References

1) Koomey, J.G. ,"Worldwide electricity used in data centres", Environ.Res. Lett. , 2008
2) Alliance to Save Energy (ASE) and 1E, "Server Energy and Efficiency Report," 2009
3) Nikzad Babaii Rizvandi, et.al, "Multiple Frequency Selection in DVFS-Enabled Processors to Minimize Energy Consumption," in *Energy Efficient Distributed Computing*, *2011, John Wiley Publisher*
4) Nikzad Babaii Rizvandi, et.al, "Some observations on Optimal Frequency Selection in DVFS–based Energy Consumption Minimization in Clusters," *Journal of Parallel and Distributed Computing (JPDC), 2011.*
5) Nikzad Babaii Rizvandi, et.al, "Linear Combinations of DVFS-enabled Processor Frequencies to Modify the Energy-Aware Scheduling Algorithms," *CCGrid 2010*
6) Nikzad Babaii Rizvandi, et.al, "A Study on Using Uncertain Time Series Matching Algorithms in Map-Reduce Applications", Concurrency and Computation: Practice and Experience, 2012
7) Nikzad Babaii Rizvandi, et.al, "Network Load Analysis and Provisioning for MapReduce Applications", *PDCAT-2012*
8) "Data-Intensive Workload Consolidation on Hadoop Distributed File System", *Grid'12*
9) Nikzad Babaii Rizvandi, et.al, "On Modelling and Prediction of Total CPU Utilization for MapReduce Applications", *ICA3PP 2012*