HP-DAEMON: *H*igh *P*erformance *D*istributed *A*daptive *E*nergy-efficient *M*atrix-multiplicati*ON*

Li Tan[1], Longxiang Chen[1], Zizhong Chen[1], Ziliang Zong[2], Rong Ge[3], and Dong Li[4]

[1]*University of California, Riverside*
[2]*Texas State University-San Marcos*
[3]*Marquette University*
[4]*Oak Ridge National Laboratory*

UCR UNIVERSITY OF CALIFORNIA, RIVERSIDE

ICCS'14, Cairns, Queensland, Australia
June 11, 2014

Power Management in HPC via DVFS

- Power and energy consumption of high performance computing is a growing severity → *operating costs* and *system reliability*.

- Dynamic Voltage and Frequency Scaling (DVFS)
  - voltage/frequency ↓ → power ↓ → energy efficiency
  - Peak CPU performance is *not necessary* when slack exists: load imbalance, network latency, communication delay, memory and disk access stalls, etc.

Overhead on Employing DVFS

- Frequency Switching Delay
  - DVFS is implemented via modifying *in-memory* CPU frequency configuration files at OS level → high memory access overhead if *too many* DVFS switches

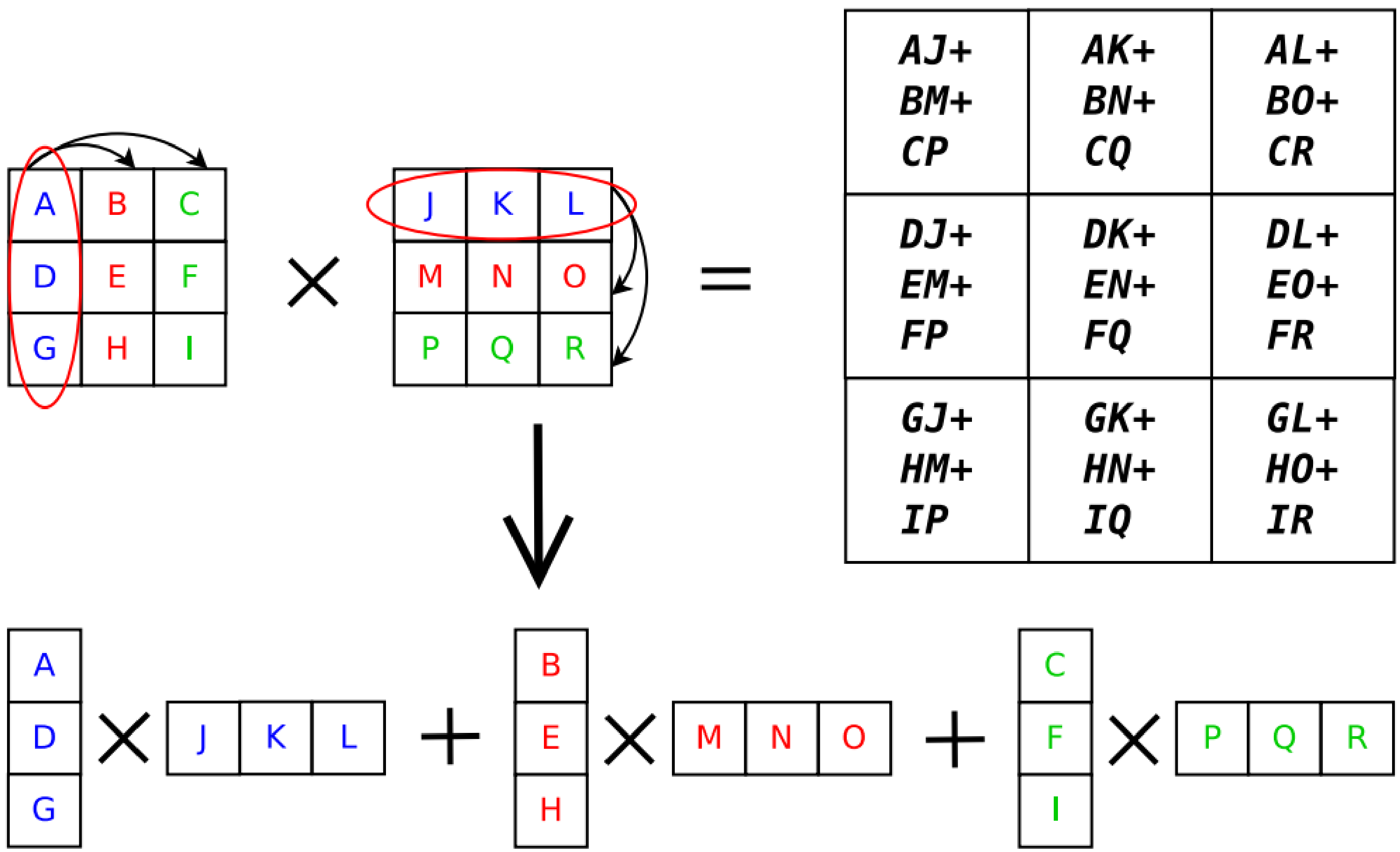
- Frequency Transition Latency
  - CPU will stay in use of the old frequency while the newly-set frequency does not *take effect*
  - Per frequency switch:
    - 100 μs for AMD Athlon processors
    - 38 μs for AMD Opteron 2380 processors (in this work)

Our Strategy

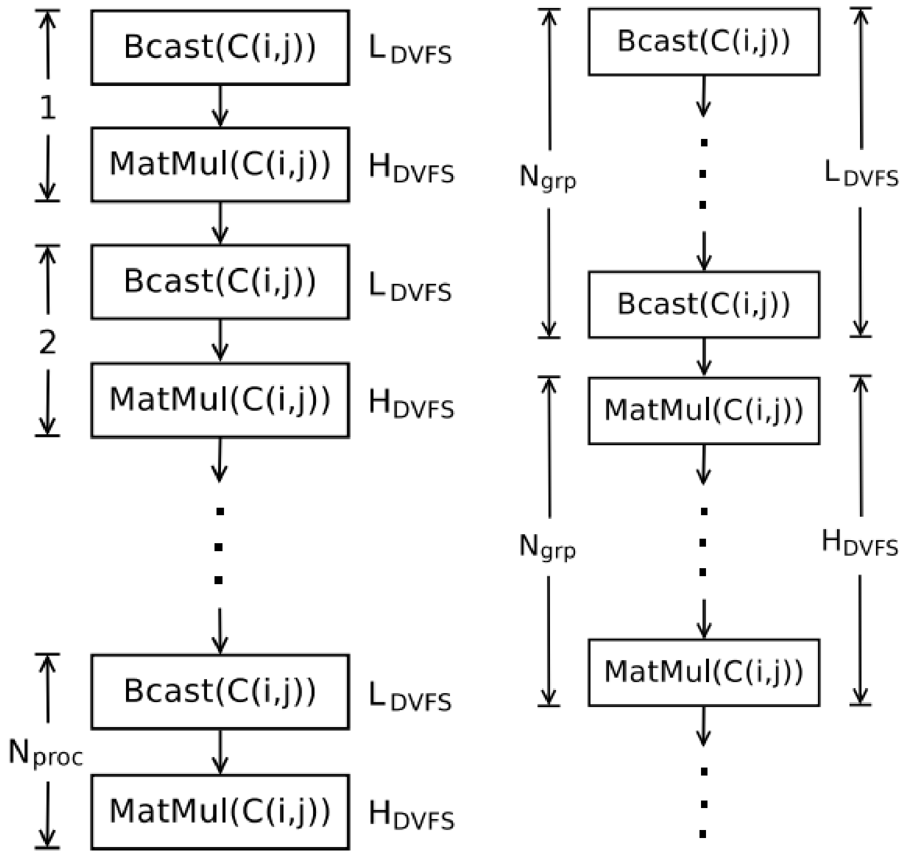
- Concern possible *non-negligible* costs on DVFS
  - Adaptively switch frequency to limits the number of DVFS switches by *grouping* comm./comp., at the cost of a user-specified memory overhead threshold

- Improve performance of communication
  - Leverage a high performance communication scheme for fully exploiting network bandwidth via *tuning chunk size* of pipeline broadcast

Distributed Matrix Multiplication Algo.



Two DVFS Scheduling Strategies in DAG



(a) $DAG_{classic}$ (b) DAG_{daemon}

Memory-aware Grouping Mechanism

- Grouping Broadcasts/Multiplications
  - For each process, keep several (i.e., N_{grp}) sub-matrices of *A* and *B* from broadcasts in memory
  - Do N_{grp} matrix multiplications later *at a time*
  - Memory overhead from grouping is *restricted* to a user-specified threshold T_{mem}

$$8 \times \left(\frac{N}{N_{proc}}\right)^2 \times 2 \times N_{grp} \times N_{proc} \leq S_{mem} \times T_{mem}$$

$$\text{subject to } 1 \leq N_{grp} \leq N_{proc}$$

  - Number of DVFS switches is effectively reduced via memory-aware grouping by a factor of N_{grp}

Memory-aware Grouping Algorithm

Algorithm 1 *Adaptive Memory-aware DVFS Scheduling Strategy*

SetDVFS(N, N_{proc})

$S_{mem} \leftarrow$ GetSysMem()
$T_{mem} \leftarrow$ GetMemTshd()
$unit \leftarrow N/N_{proc}$
$N_{grp} \leftarrow S_{mem} \times T_{mem}/(8 \times unit^2 \times 2)$
$nb \leftarrow N_{proc}/N_{grp}$
foreach $i < nb$ **do**
  if (IsBcast() $\&\&$ $freq$!= L_{DVFS}) **then**
    SetFreq(L_{DVFS})
  end if
  if (IsMatMul() $\&\&$ $freq$!= H_{DVFS}) **then**
    SetFreq(H_{DVFS})
  end if
end for

Energy Efficiency Analysis

- Compare the basic strategy with *DAEMON*
  - N_{grp} determines the *extent* of grouping and thus the DVFS overhead *reduced*

$$e_{DVFS}^{basic} = e_{DVFS}^{unit} \times 2 \times N_{proc} \times \frac{N/N_{proc}}{BS} \times N_{proc}^2$$

$$e_{DVFS}^{daemon} = e_{DVFS}^{unit} \times 2 \times \frac{N_{proc}}{N_{grp}} \times \frac{N/N_{proc}}{BS} \times N_{proc}^2$$

$$E_{def} = \frac{e_{DVFS}^{basic}}{e_{DVFS}^{daemon}} = N_{grp}$$

$$E_{sav} = e_{DVFS}^{basic} - e_{DVFS}^{daemon}$$

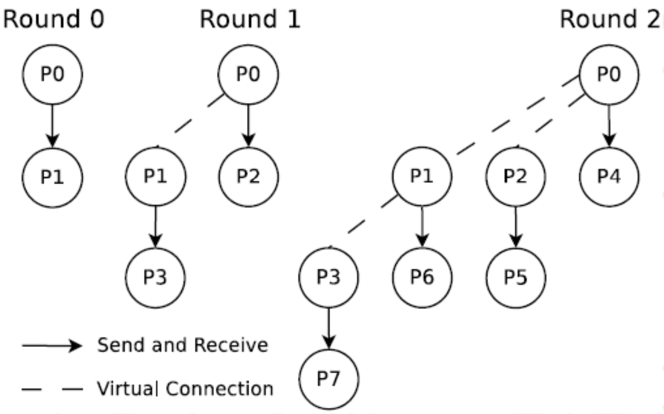
$$= e_{DVFS}^{unit} \times 2 \times \frac{N}{BS} \times N_{proc}^2 \times \left(1 - \frac{1}{N_{grp}}\right)$$

High Performance Communication

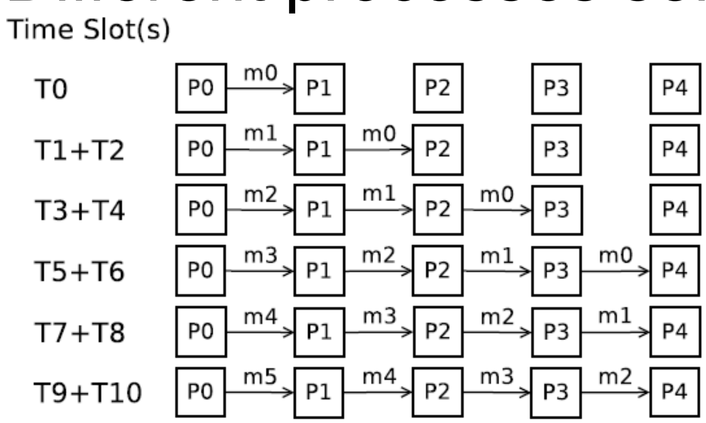
- Binomial Tree Broadcast
  - In round **j**, the number of senders/receivers is **2^j**

$$T_B = T_b \times \log P,$$

$$\text{where } T_b = T_s + \frac{S_{msg}}{BD}$$



- Pipeline Broadcast
  - Different processes send/receive message chunks



$$T_P = T_p \times (n + P - 1),$$

$$\text{where } T_p = T_s + \frac{S_{msg}/n}{BD}$$

Performance Efficiency Analysis

- Binomial Tree and Pipeline Broadcast
  - By substitution, we can obtain the simplified communication time complexity of two broadcasts

$$T_B \approx \frac{S_{msg}}{BD} \times \log P$$

$$T_P \approx \frac{S_{msg}}{BD} \times \left(1 + \frac{P-1}{n}\right)$$

  - Both T_B and T_P scale up as P increases
  - T_P decreases as n increases, which can make it finally smaller than T_B → Pipeline broadcast can *outperform* binomial tree broadcast via *tuning* n

Implementation

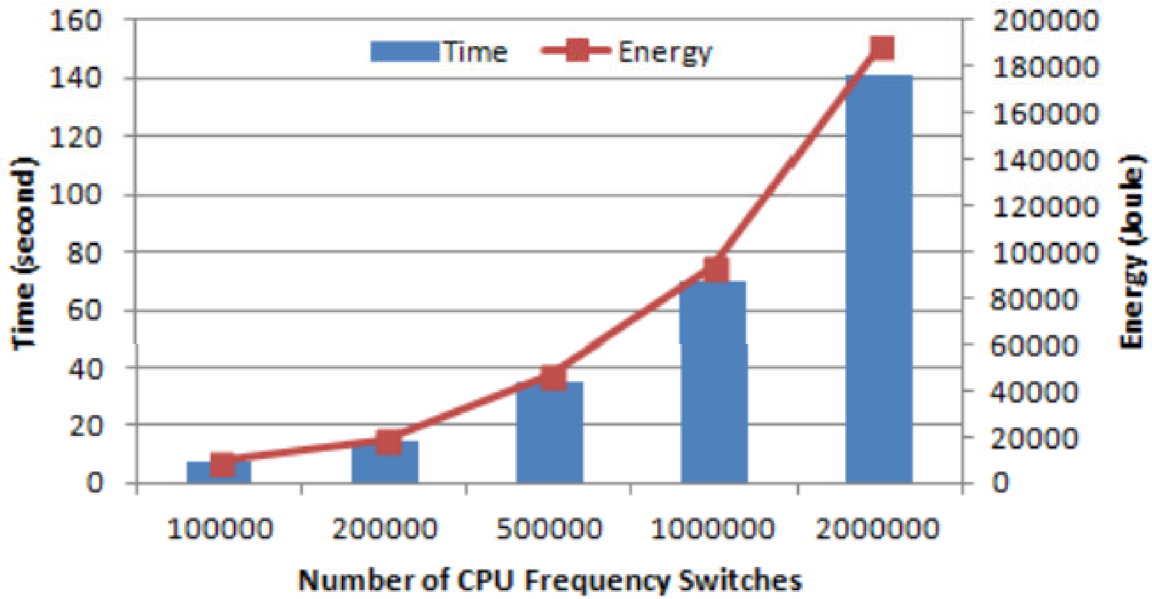
- *HP-DAEMON* → *HP* + *DAEMON* (perf. + energy)
  - *HP* : Pipeline broadcast with tuned chunk size
  - *DAEMON* : Adaptive memory-aware grouping
    (we also implemented the basic DVFS strategy)
  - *Target Application* : Distributed matrix multiplication
- Rewrite pdgemm() from ScaLAPACK/DPLASMA
  - *Computation* : Employ highly tuned ATLAS dgemm()
  - *Communication* : Employ core tiling topology +
    Non-blocking highly tuned pipeline broadcast
  - *Programming Interface* : Same as ScaLA./DPLASMA

Evaluation

- Hardware Configuration

Cluster	HPCL (Energy+Perf.)	Tardis (Perf. Only)
System Size	8	16
Processor	2 x Quad-core (totalling 64 cores)	2 x 16-core (totalling 512 cores)
Memory	8 GB RAM	64 GB RAM
Network	1GB/s Ethernet	40GB/s Infiniband
Power-aware?	Yes	No
CPU Frequency	0.8, 1.3, 1.8, 2.5 GHz	N/A
Power Meter	PowerPack	N/A

Overhead on Employing DVFS

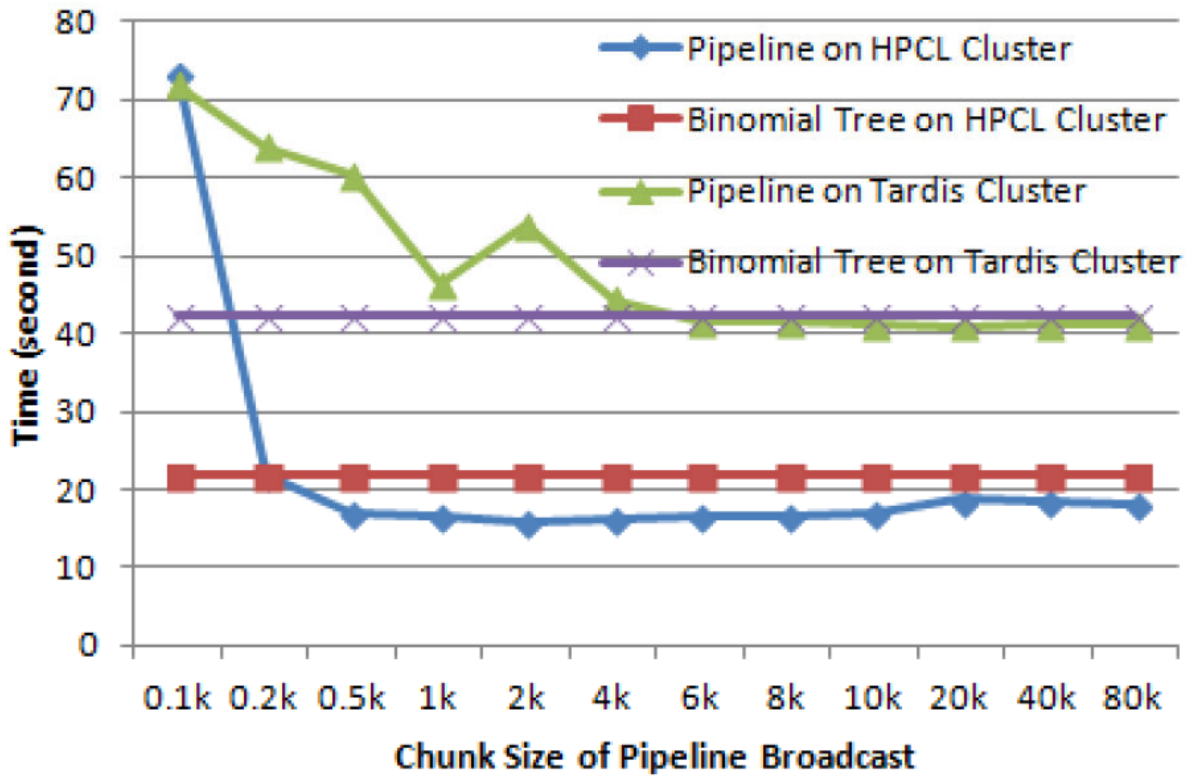


Memory Overhead Trade-off

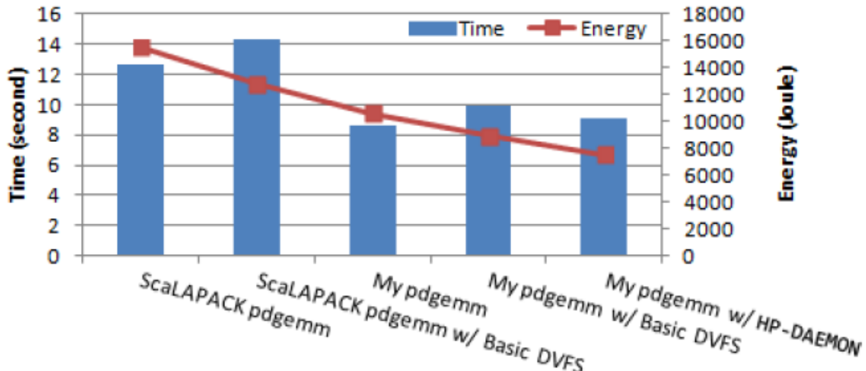
Table 3: Memory Overhead Thresholds for Different Matrices and N_{grp}.

Global Matrix Size	N_{grp}	Theoretical Additional Memory Overhead	Observed Total Memory Overhead
7680	2	3.2%	6.4%
	4	6.4%	8.8%
	8	12.8%	14.4%
10240	2	4.8%	10.4%
	4	9.6%	16.0%
	8	19.2%	25.6%
12800	2	8.0%	16.0%
	4	16.0%	24.0%
	8	32.0%	40.0%
15360	2	11.2%	23.2%
	4	22.4%	35.2%
	8	44.8%	57.6%
17920	2	16.0%	28.0%
	4	32.0%	43.2%
	8	64.0%	78.4%

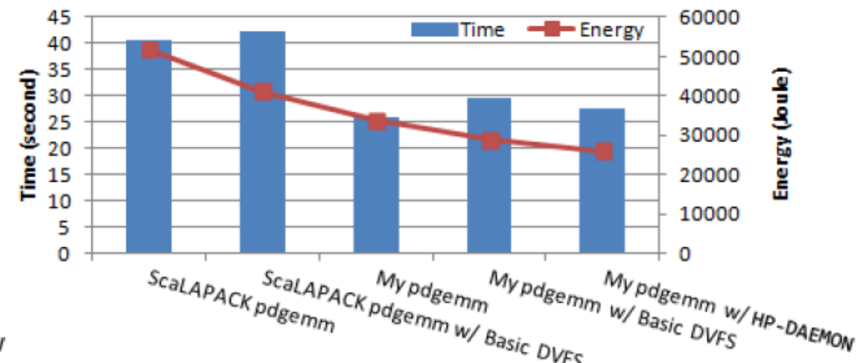
Performance Gain via Pipeline Bcast



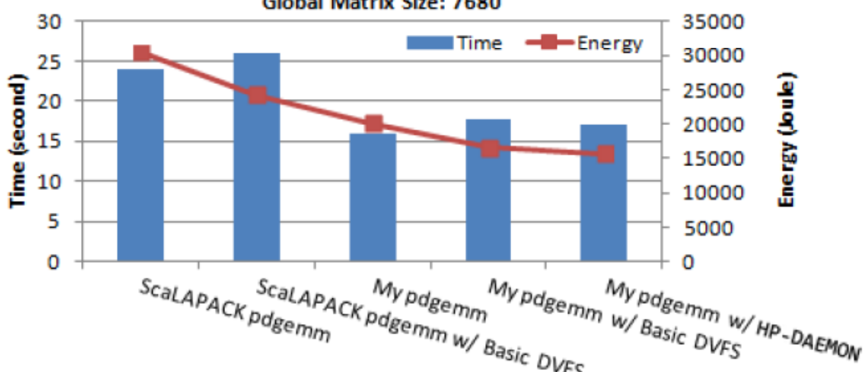
Performance Loss and Energy Savings



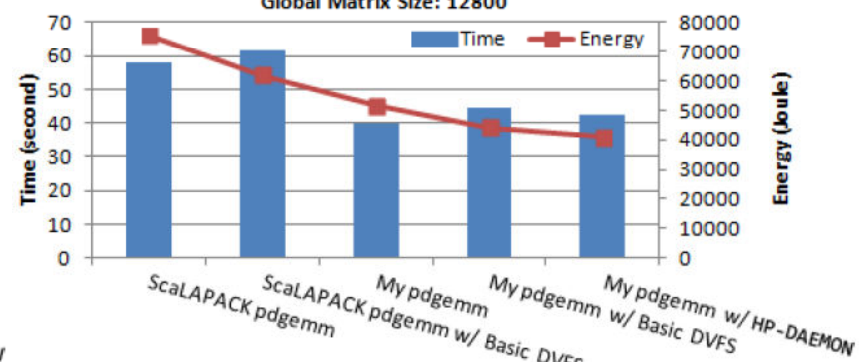
Global Matrix Size: 7680



Global Matrix Size: 12800

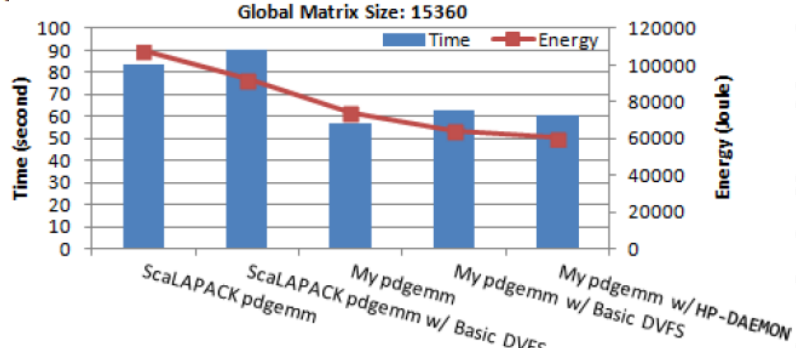


Global Matrix Size: 10240



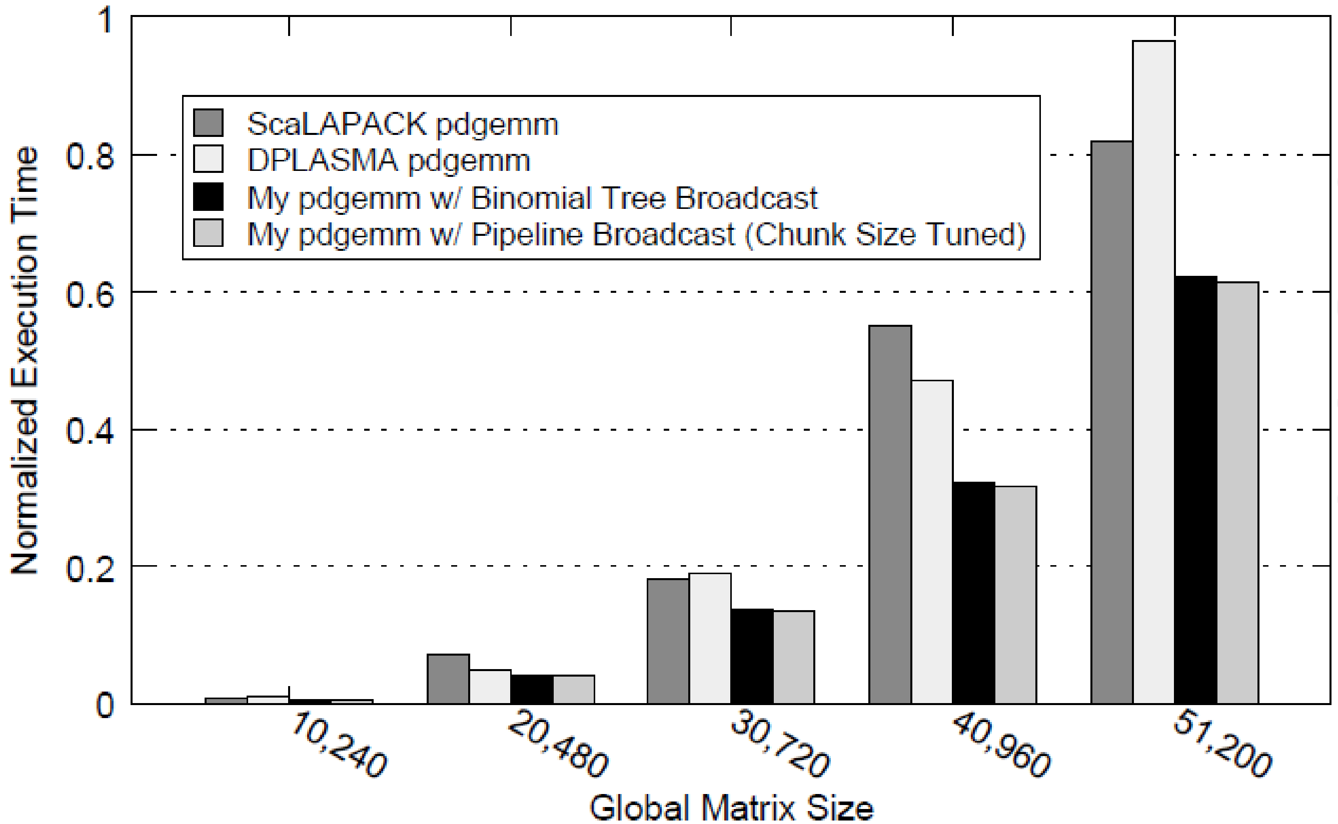
Global Matrix Size: 15360

Global Matrix Size: 15360



Global Matrix Size: 17920

Performance Gain on Fast Network



Conclusions

- Adaptive Memory-aware DVFS Scheduling
  - Minimize the overhead on employing DVFS by reducing the number of frequency switches
  - Grouping broadcasts/multiplications within user-specified memory overhead threshold
- High Performance Pipeline Broadcast
  - Boost perf. of communication via tuning chunk size
- Integrated Approach
  - The optimal energy and perf. efficiency overall is thus achieved on two clusters vs. ScaLAPACK/DPLASMA