Scheduling Sparse Numerical Computations For Energy & Performance Efficiencies

Padma Raghavan

Computer Science and Engineering
Institute for CyberScience
The Pennsylvania State University

Research Supported by NSF, DoD and IBM

Scheduling 2009, Knoxville May 13-15, 2009

Outline

- I. HPC architectures
  - Impacts on algorithm & s/w design—opportunities for scheduling
- II. Energy-Aware Scaling
  - Increasing efficiency: Examples of algorithm & s/w redesign through scheduling
- Summary

Scheduling

I. Scheduling used loosely to indicate
   - Combinations of data staging and task scheduling; turning load imbalance into energy savings; core-thread assignments for energy-saving

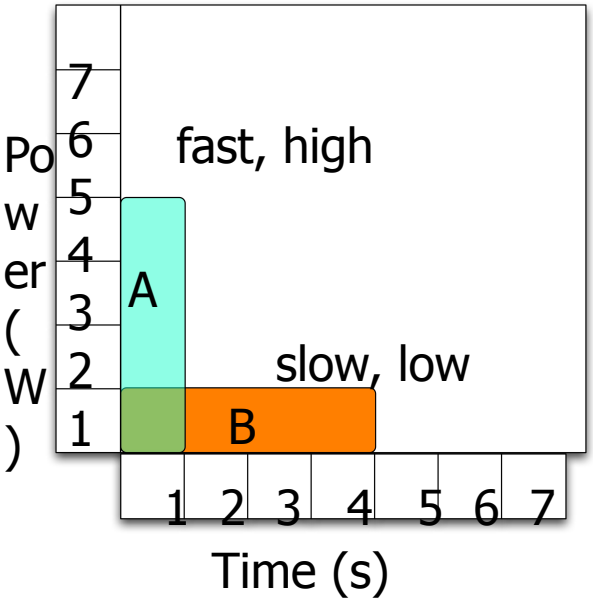
II. Several instances could potentially be formulated in more traditional forms such as
   - Multi-objective dynamic scheduling
   - Master thread assigning pool of tasks
   - All with resource, capacity constraints

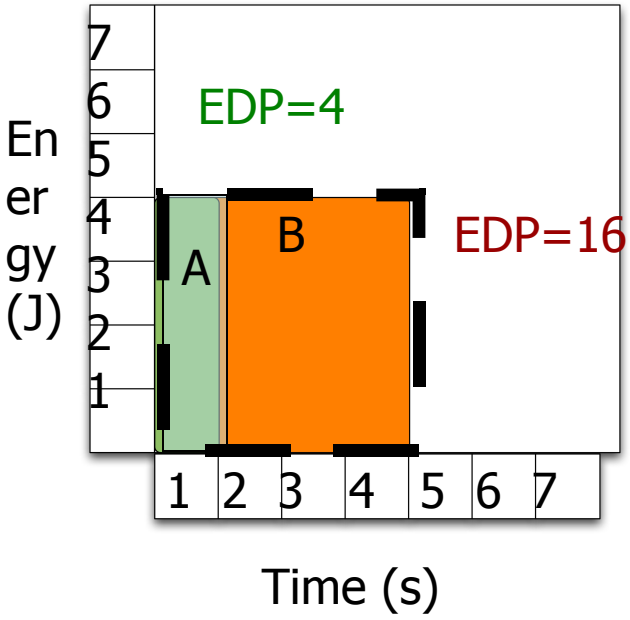
Voltage/frequency Scaling

- Reducing supply voltage (Vdd) decreases dynamic energy
  - Dynamic Energy ~ C*f*Vdd^2
- Frequency (f) decreases with supply voltage, leading to longer execution time
- Leakage Energy ~ Ileak * Vdd / f
- EDP = Energy * Delay
  ~ (Dynamic Energy + Leakage Energy) / f

Measuring Energy Efficiency

Same code on two different systems A and B





Equal energy (PDP) does not differentiate A from B

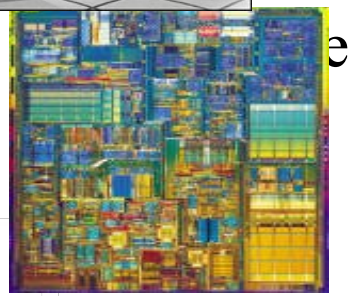
Energy**D**elay**P**roduct (**E**nergy X **T**ime) is lower for faster system A

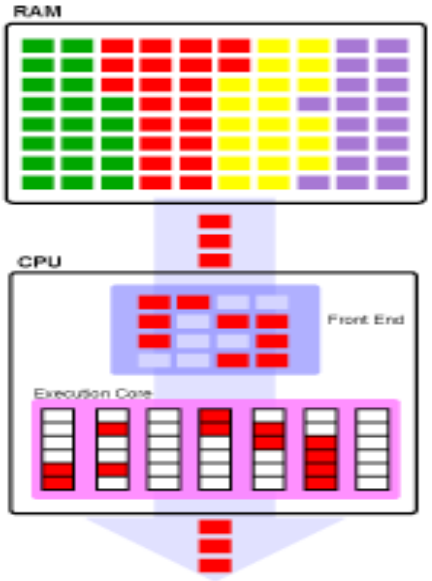
I: Exascale Architectures

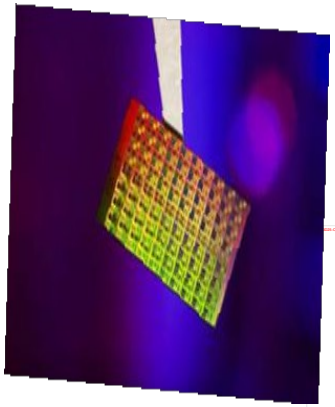
- Then, now and beyond
  - From fast, hot …
  - To parallel, cooler
  - To billion-way parallel, heterogeneous, unreliable

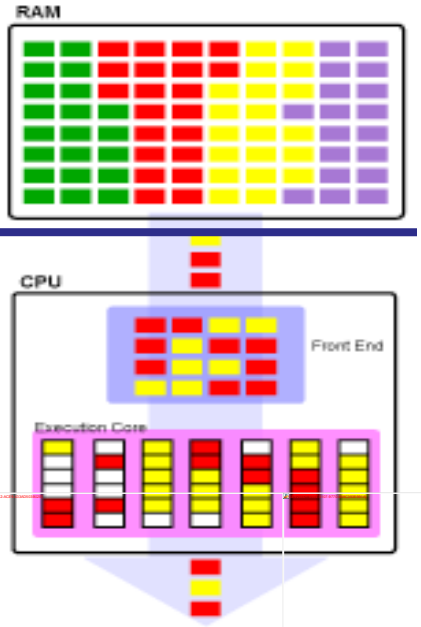
Toward Exascale

Future systems will consist of millions of nodes









ILP Multi node Multi core Multithreaded

P = P x 4 P = P x 4 x 100 x 4

150MW 150MW 150MW

x 1,000,000

150MW

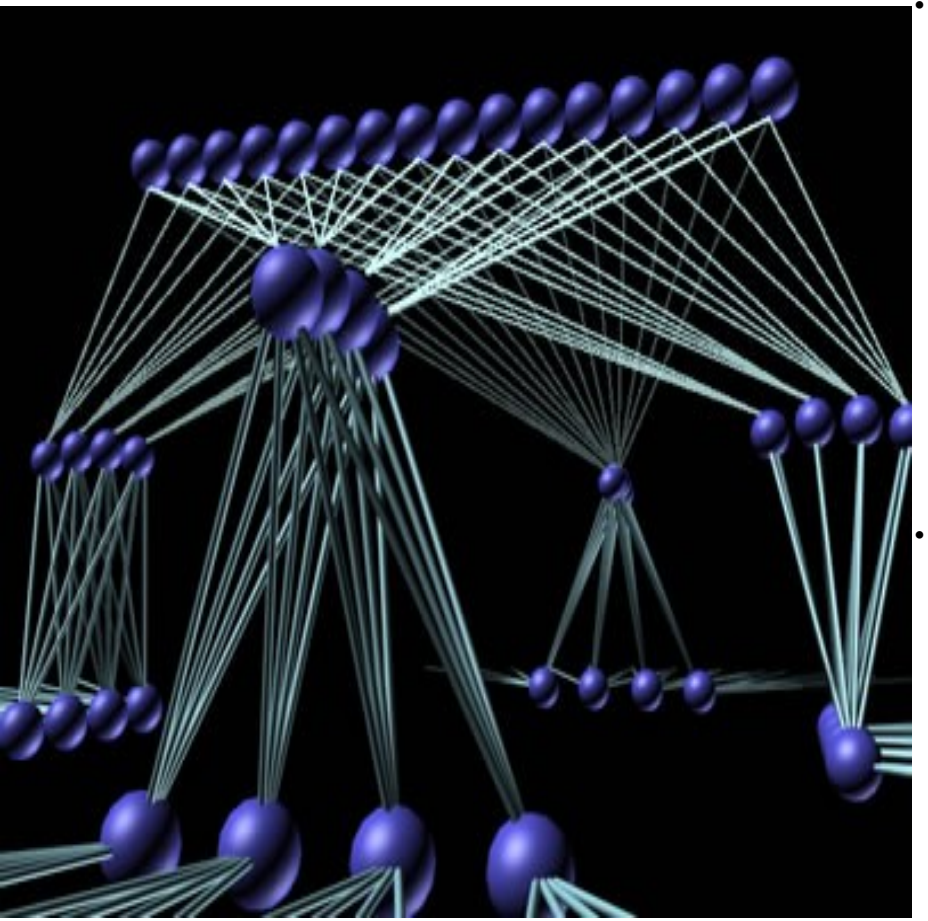
Fixed power per chip

More cores, threads per chip

billion-way parallelism

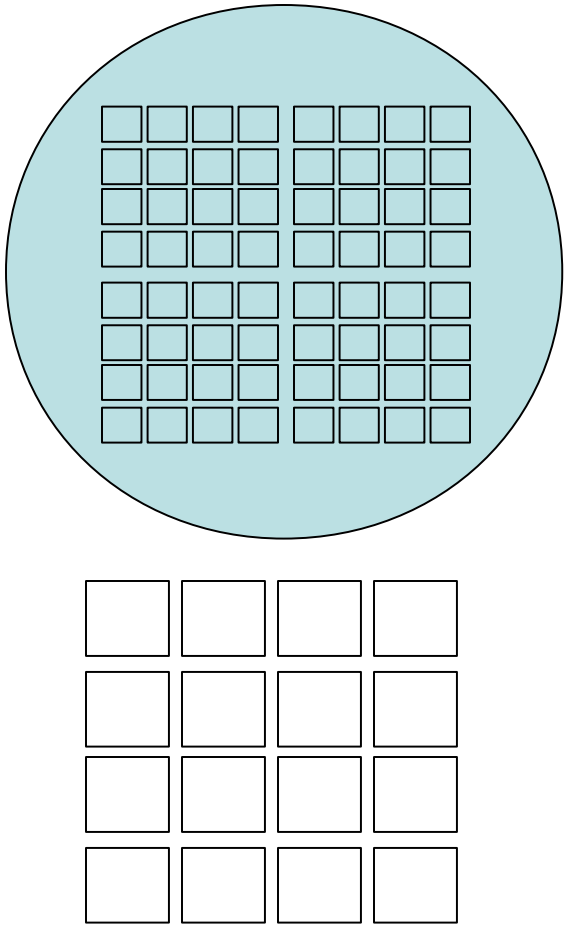
Demands algorithm redesign to increase efficiency

Importance of Network Power



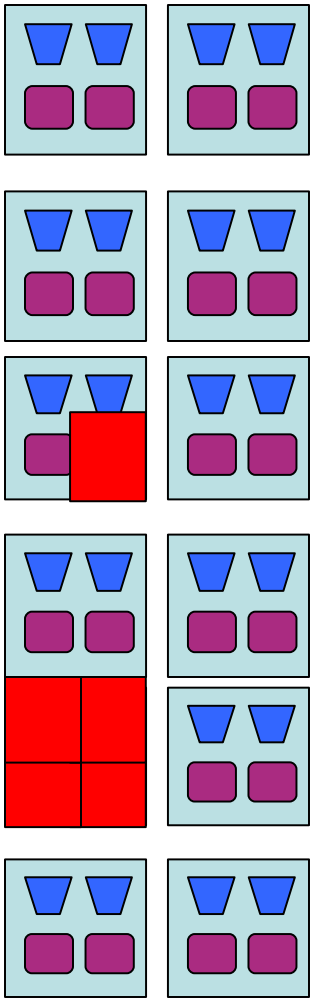
- Network power will increase in importance
  - As high as 60-70% system
  - DVFS and link throttling options to save energy
- Can messaging be scheduled to reduce energy without performance impacts while link throttling ?

Process Variability



- Manufacturing is imperfect
- Die for 4 chips@ 16 cores
  - Top fast, high leak
  - Bottom slow, low leak
  - Variations within chip
- Reorganize computations to model variations
- Schedule and load balance for performance and energy
- Algorithms/software will have to model these variations

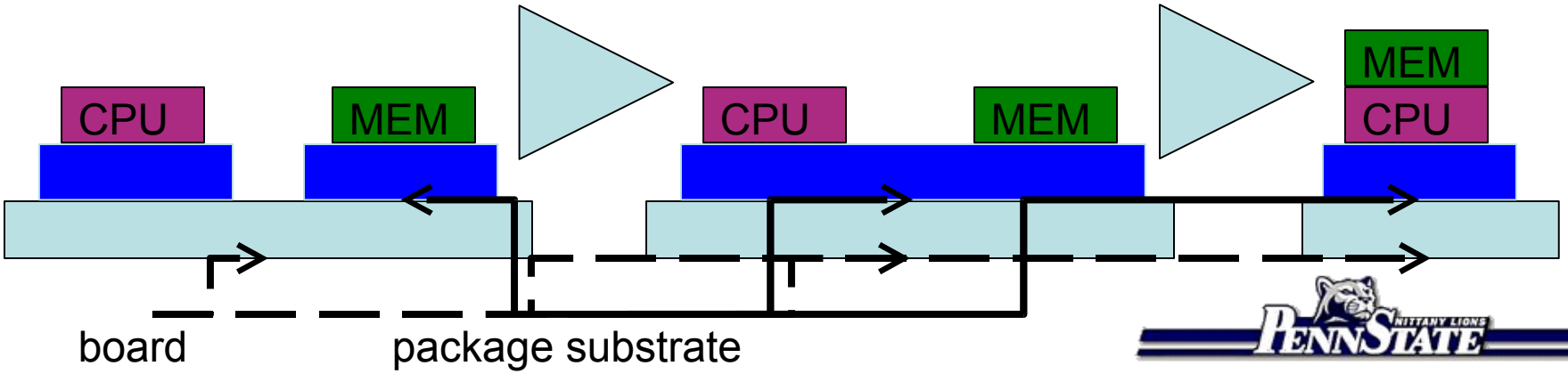
Failures & Soft-Errors



- Components will fail in 100 core chips
  - Not cost effective to throw out chip
- Use cores in diminished capacity
  - Example: failure of one functional unit
  - Disable core if unusable
- Soft errors (bit flips) in low V regimes & algorithm correctness
- Algorithms/software have to adapt – opportunities for scheduling

Caches & Memory

·Caches not useful for applications w/o reuse or long reuse distances
·Alternatives such as user programmable memories (scratchpad) that are power efficient
·Options for data-staging to mask latency
·Algorithm/software reorganization, pipelining, optimizing off-chip to on-chip throughput



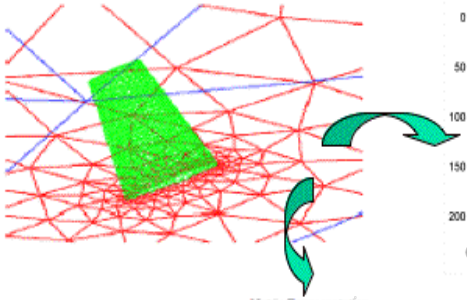
II: Energy-Aware Scaling: Focus on Efficiency

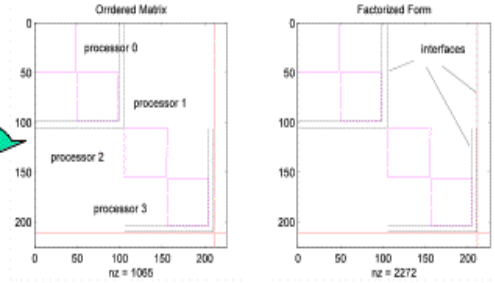
- Exploiting concurrency & managing power for O(N) sparse graph/matrix
  - Algorithm/library design for parallelism and dynamic adaptivity for energy efficiency
    - Across nodes & network
    - At node
- Results using simulators
  - Accepted methodology in computer architecture
  - Limit-study of potential benefits

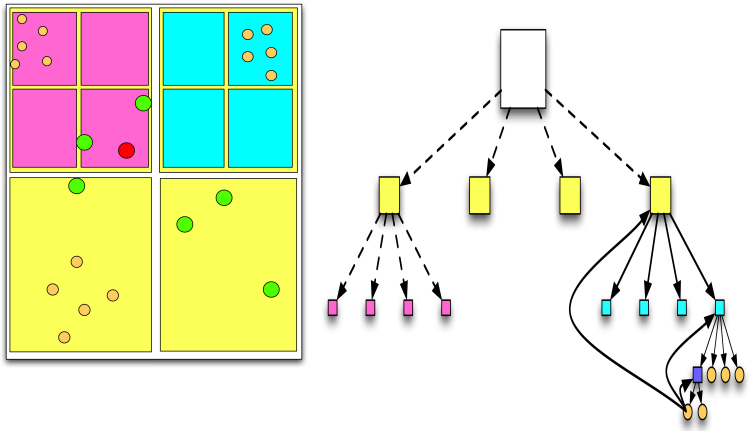
Sparse/Irregular Data Driven Computations

Interoperable, Sparse Data Structures and Transformations

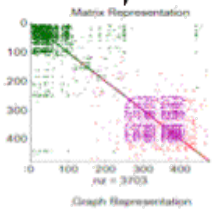
Discretizing with adaptive meshes

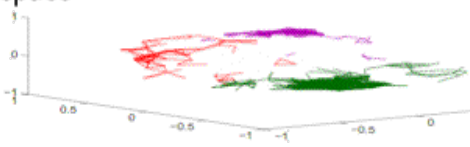






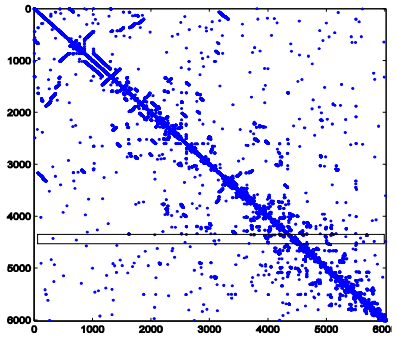
Graphs, networks, clusters in low dimensional space

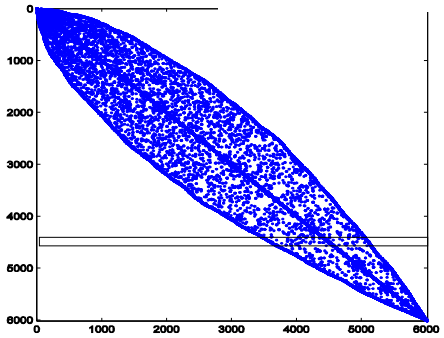




Sparse matrices, automatically ordered to reveal hierarchical structure

Sparse structures enable linear scaling of computational cost with problem size, parallelism,..





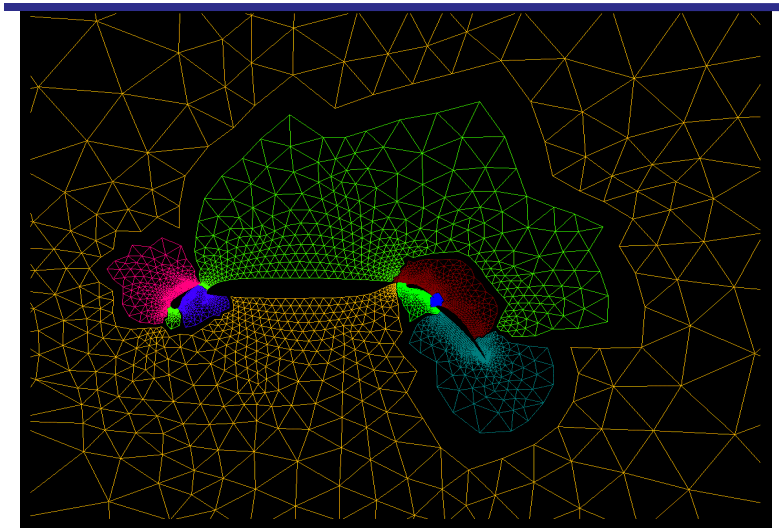
Application requirements-> algorithm selection + tuning -> H/W, S/W adaptivity

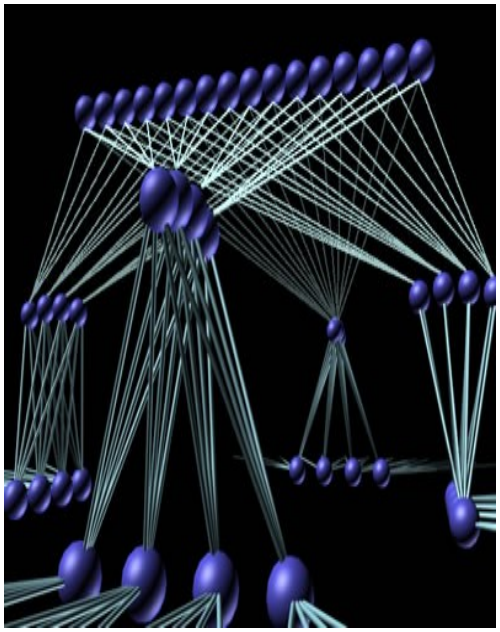
PENNSTATE NITTANY LIONS

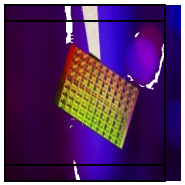
II: Energy-Aware Algorithms

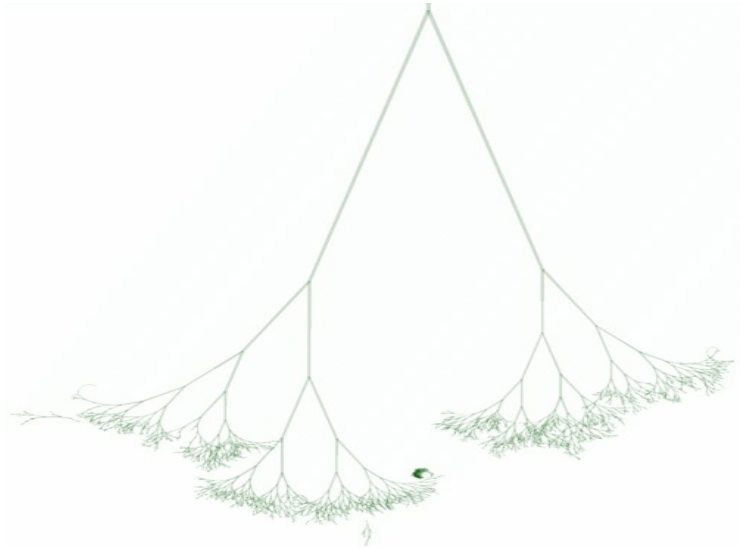
- II (a) Opportunities across nodes
  - Network energy and collective communications
  - HPPAC-IPDPS'06 – Sarah Conner, Mary Jane Irwin, P. R.

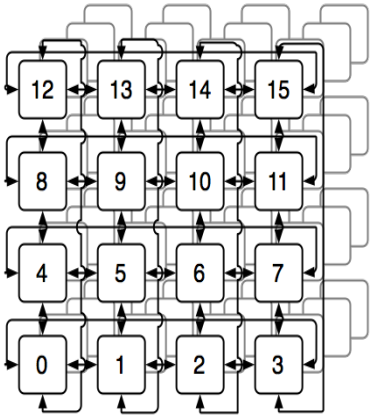
Partition and Map to MPP Nodes







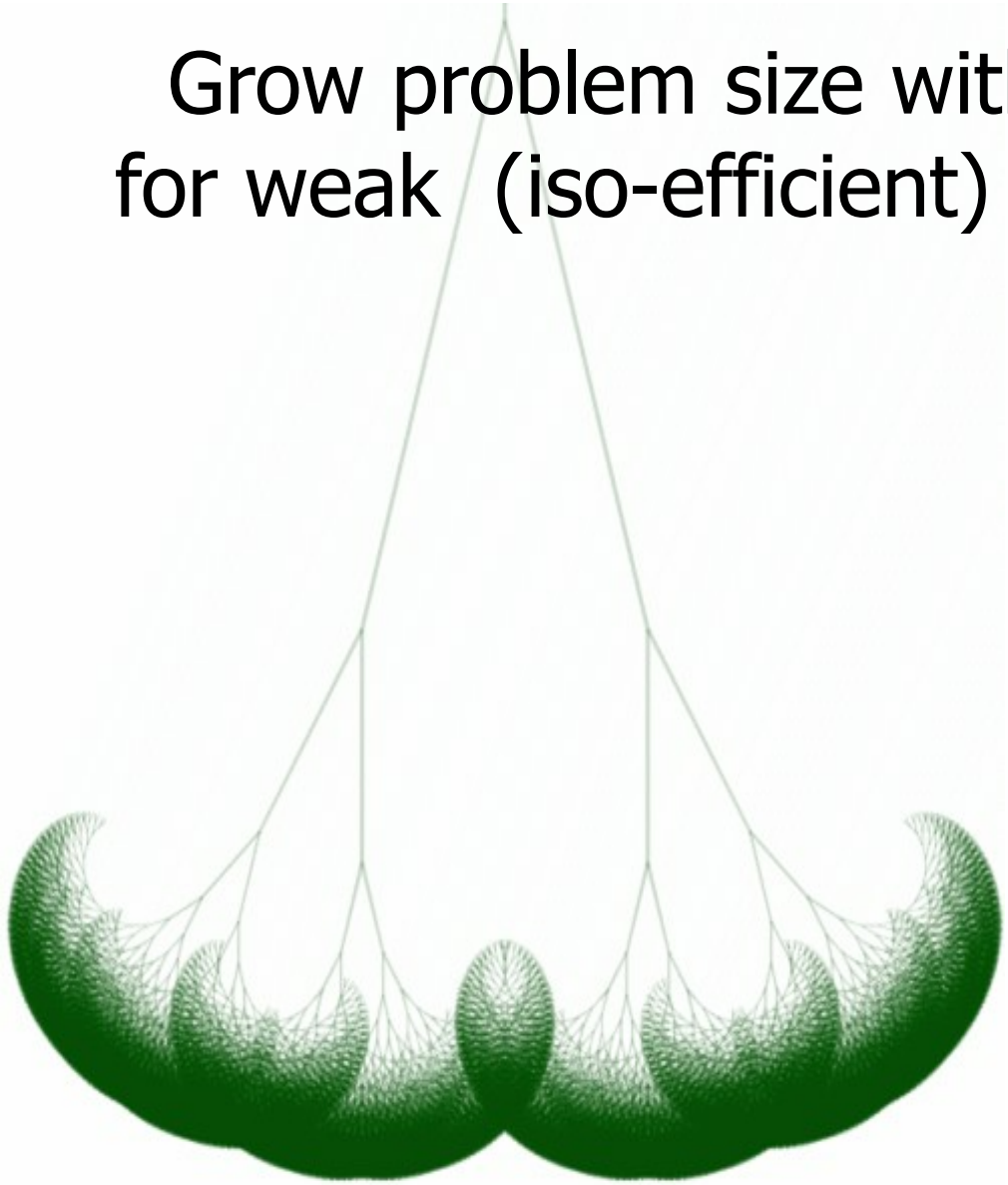


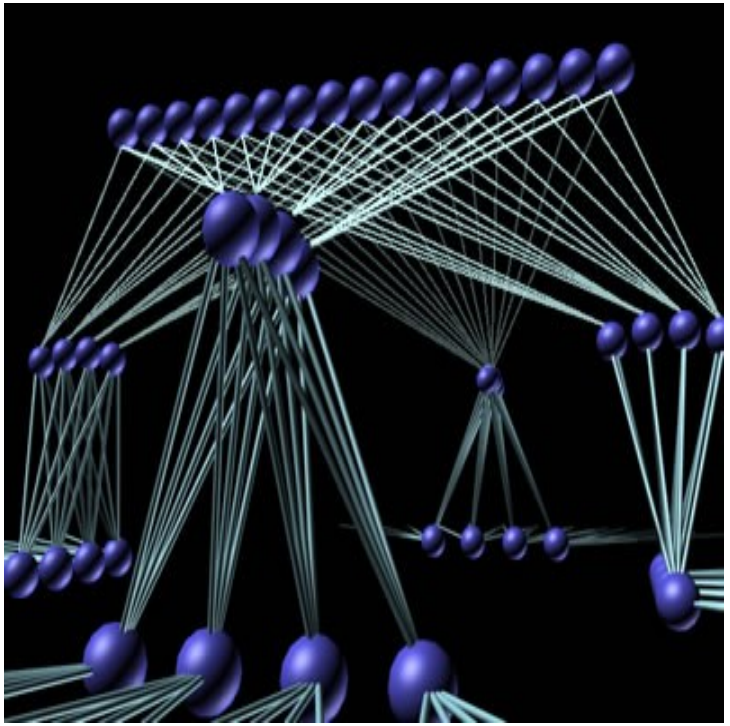


PENNSTATE NITTANY LIONS

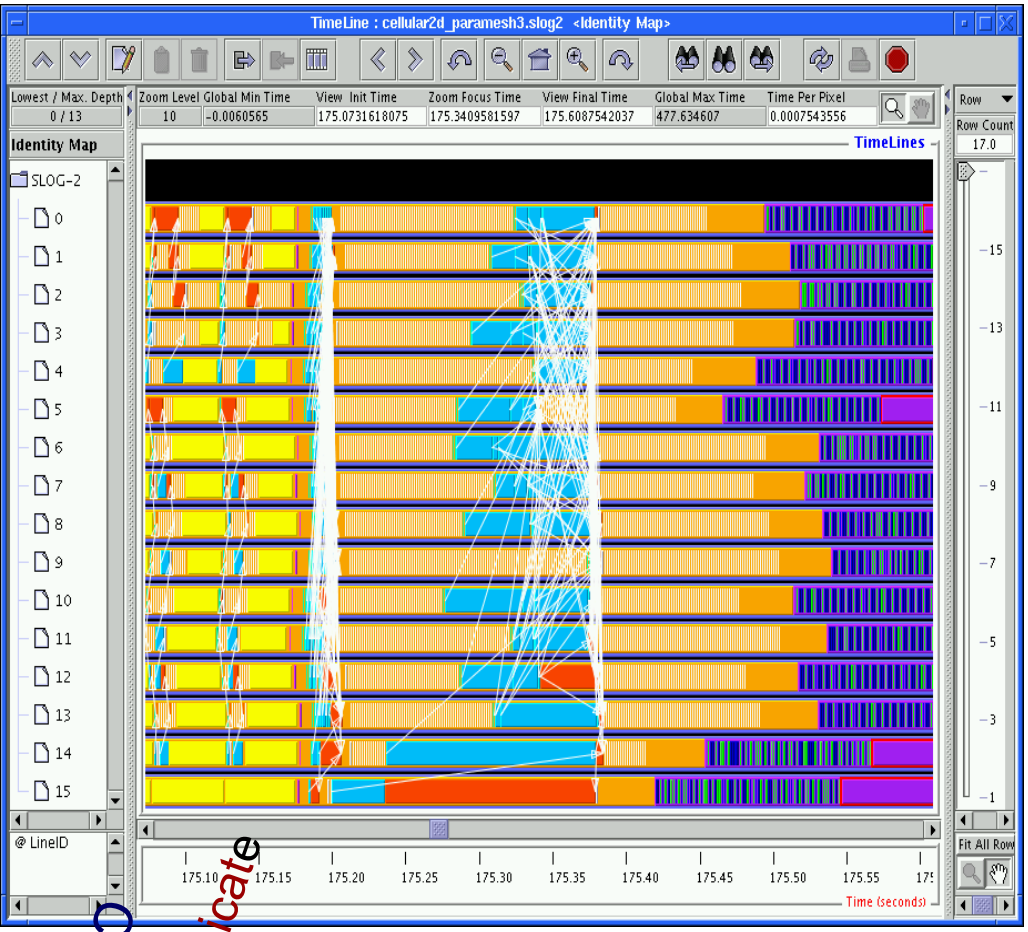
Cross Node Scaling

Grow problem size with number of nodes for weak (iso-efficient) scaling





Torus Network: Managing Power

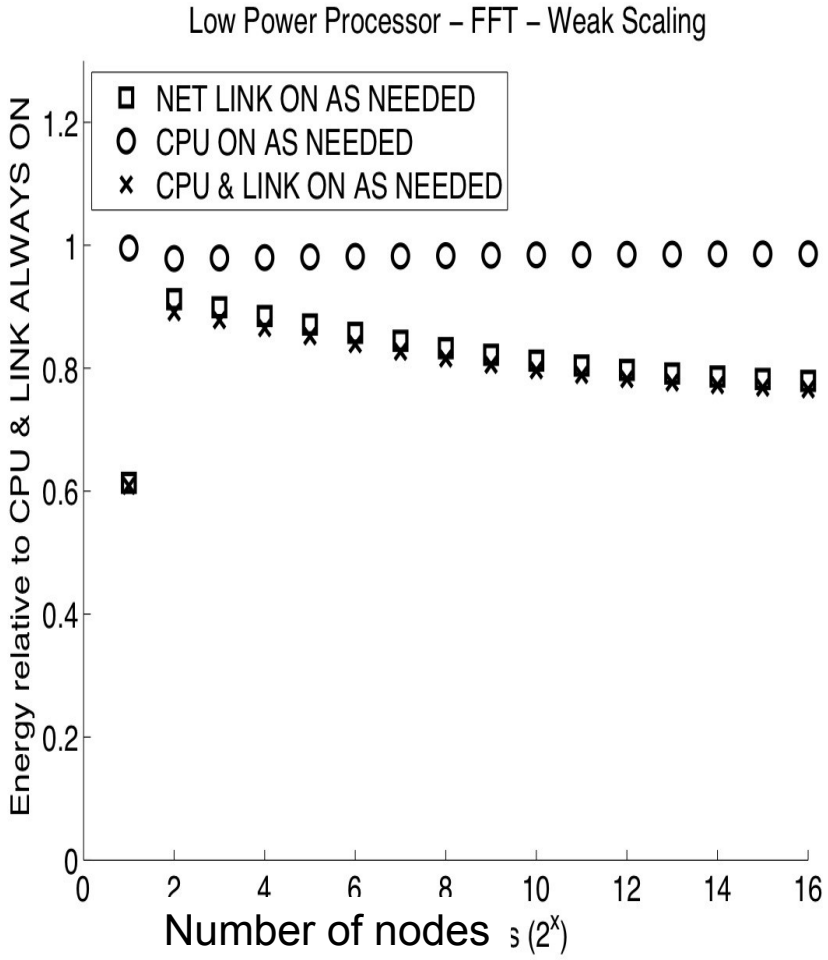


Across nodes: compute & communicate phases

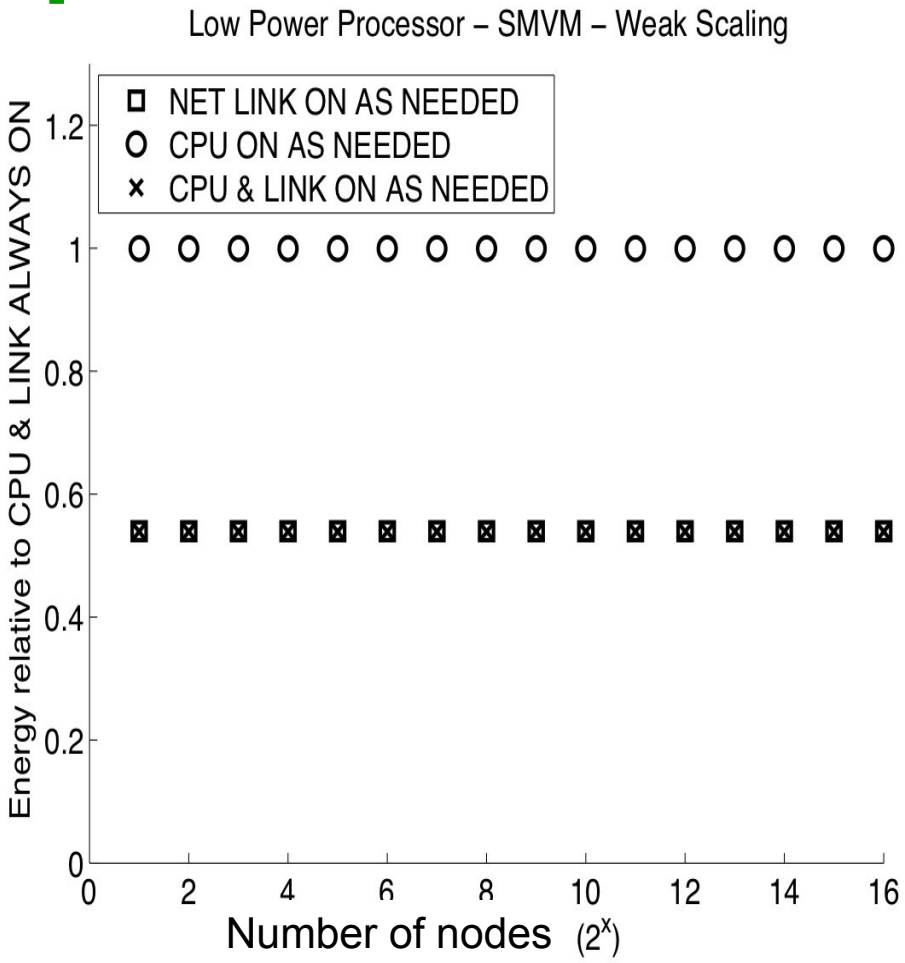
- Can network link shutdown
  - Save energy for an application?
  - Save energy even at a collective communication?

Network Energy & Weak Scaling

FFT

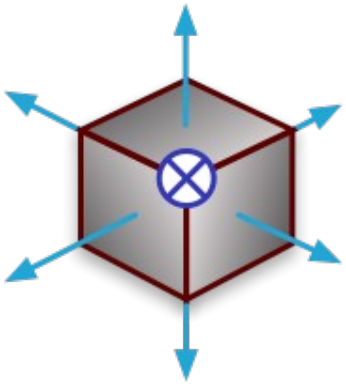


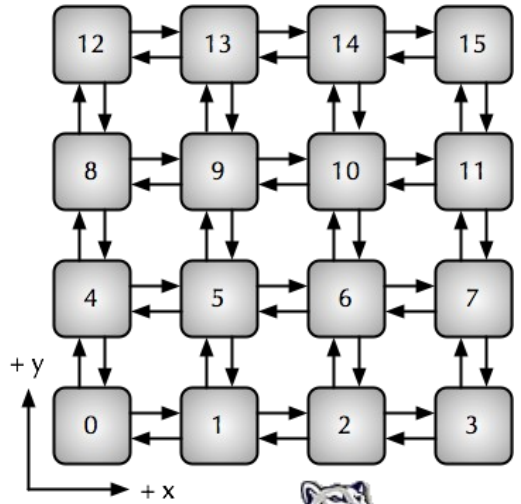
Sparse Mat-Vec



Collective Communications on a Torus Network

- Consider operations such as reduce
- Focus on energy saving opportunities
- Methods could be programmed into libraries like MPI for energy-aware scaling
- Network parameters as in BG/L

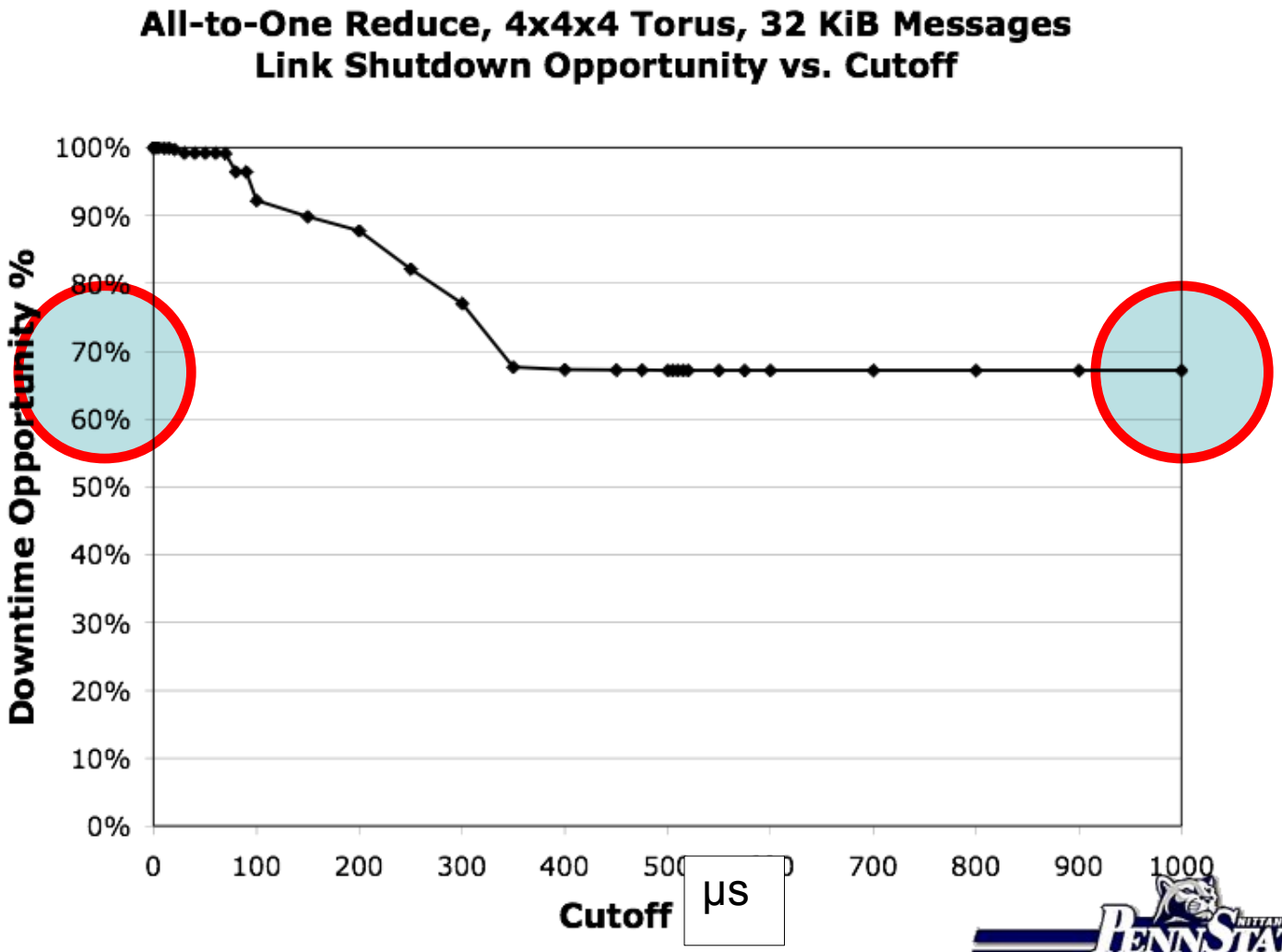




Link Shutdown Timer?

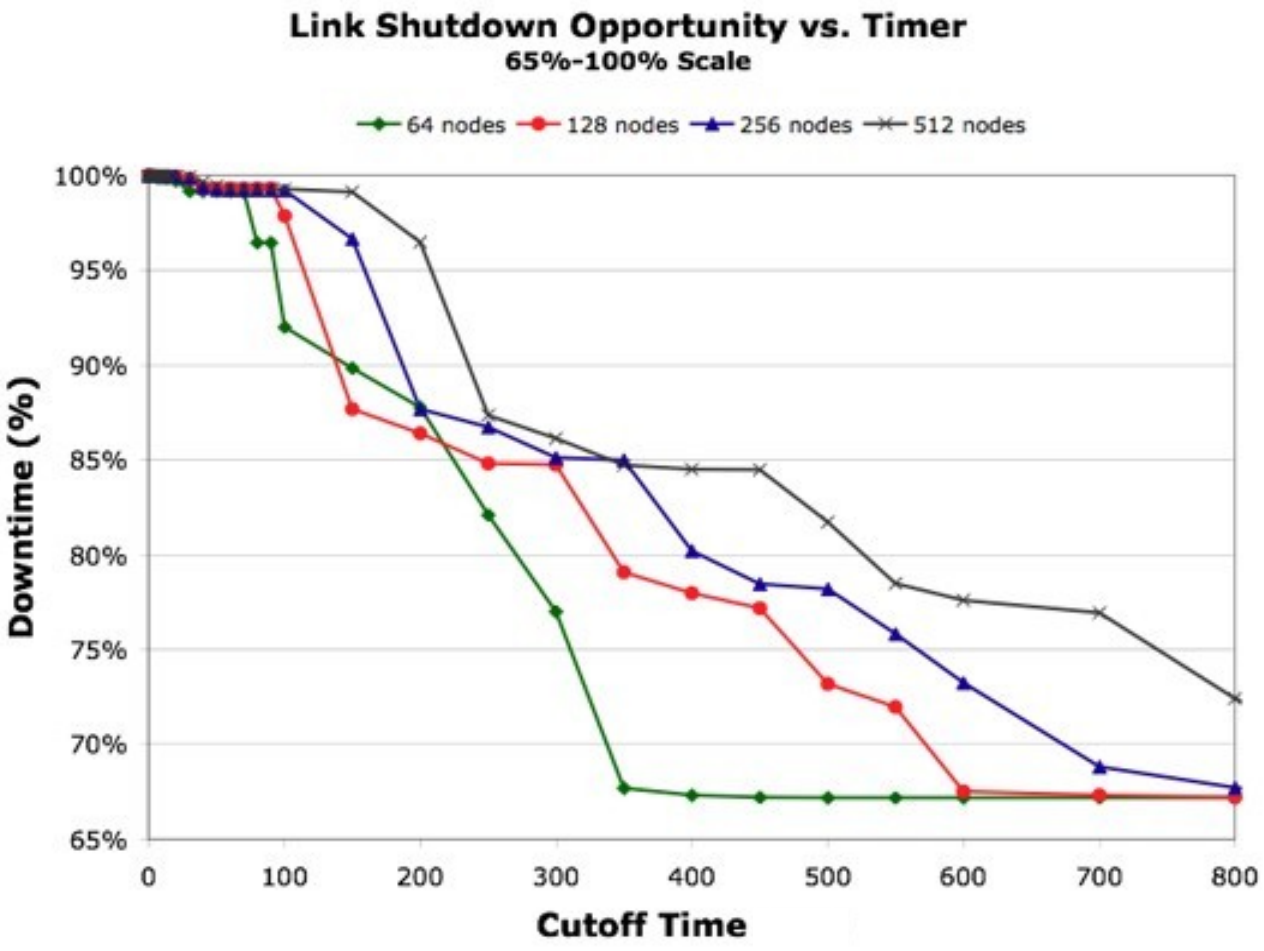
- Link shutdown: Electrically disconnecting links between nodes that aren't in use
- Includes link (cables) plus routers, buffers, etc. — Active and passive energy saved
- Cutoff timer: How long a link is inactive before it is shut off
- Wakeup latency: How long it takes to wake a link

Reduce Results



Link Shutdown in Collective Communication

Reduce Operation, 512 node torus



- Many links remain unused. For reduce, it's 66%
- Implement simple link shutdown (LS) hardware in the net
- Library code X LS hardware can utilize link shutdown
- With 100 µs over 99% shutdown is available

II: Energy-Aware Scaling

- II (b) Opportunities across nodes
  - Converting load imbalance to energy savings
  - Tree-based sparse codes
  - HPPAC-IPDPS'06 – G. Chen, K. Malkowski, M. Kandemir, P. R.

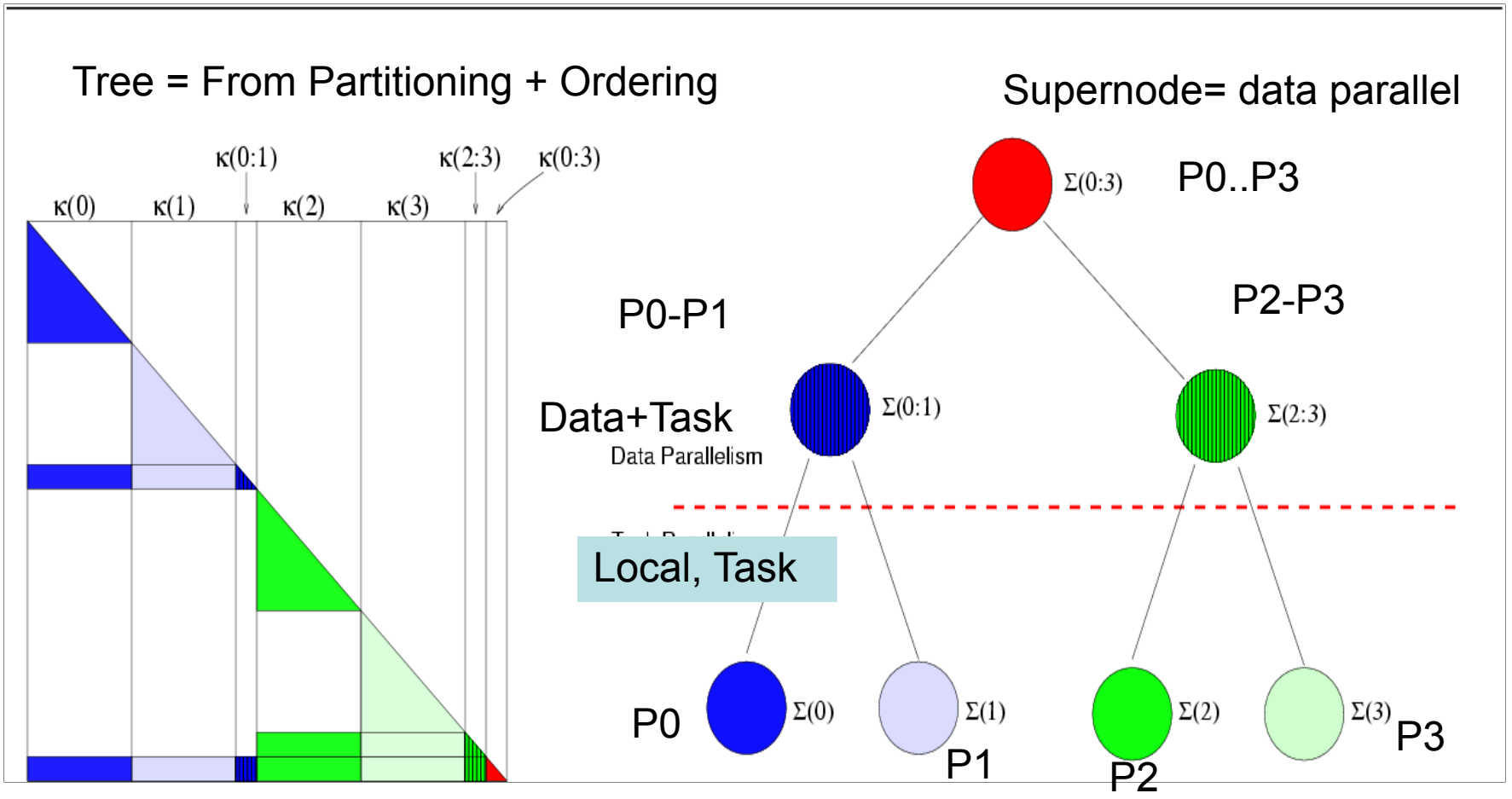
Tree-Based Parallel Multifrontal/Panel Sparse Solver

- Each tree node= dense/blocked matrix operations
- The tree dictates data-dependencies
  - Computations at a node depend only on subtree rooted at the node
  - Computation in disjoint subtrees are independent
- Task-parallelism at lower levels,data-parallelism at higher levels

- Tree is weighted with computation & communication costs
- Model can generalize to Barnes-Hut etc. ….Tree-based Hybrid Direct-Iterative Preconditioned Solvers (SIAM J. Sci. Comp, Teranishi, P.R.)

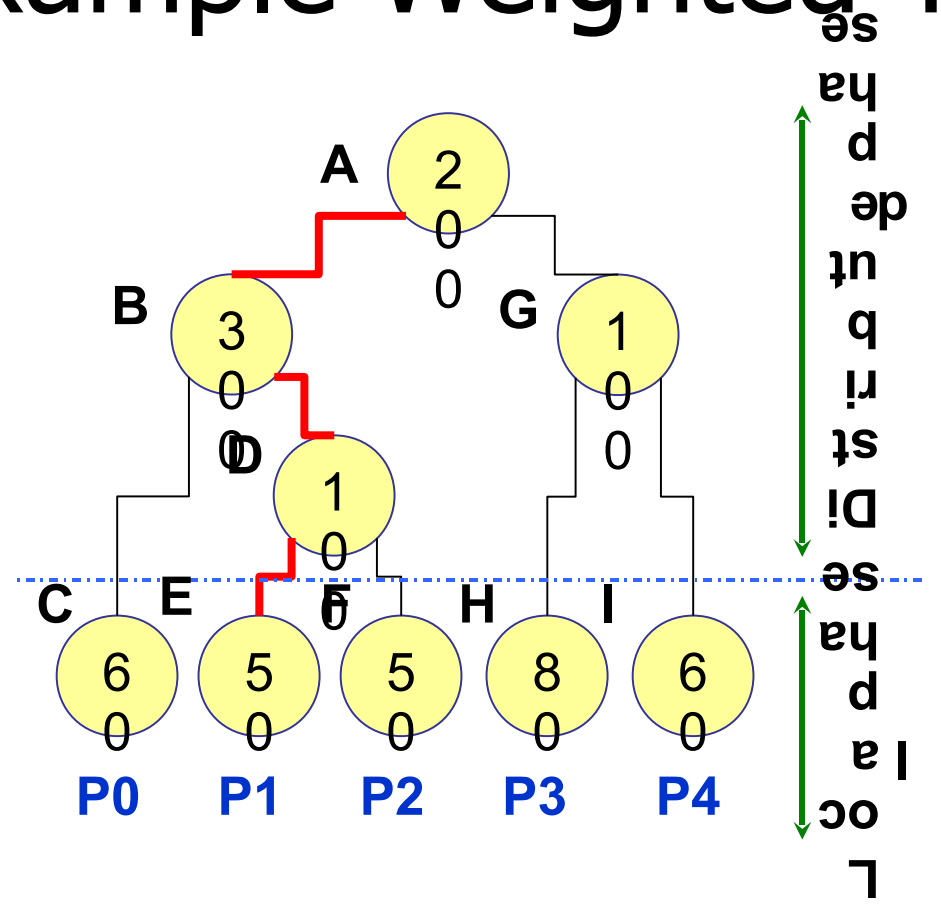
- Tree mapped to processors for load-balance
  - Even "best" scheme has imbalances of 10-30% of ideal
  - Processors and links on lightly loaded paths can be voltage

Coarse Grain: Tree-Structured Parallelism

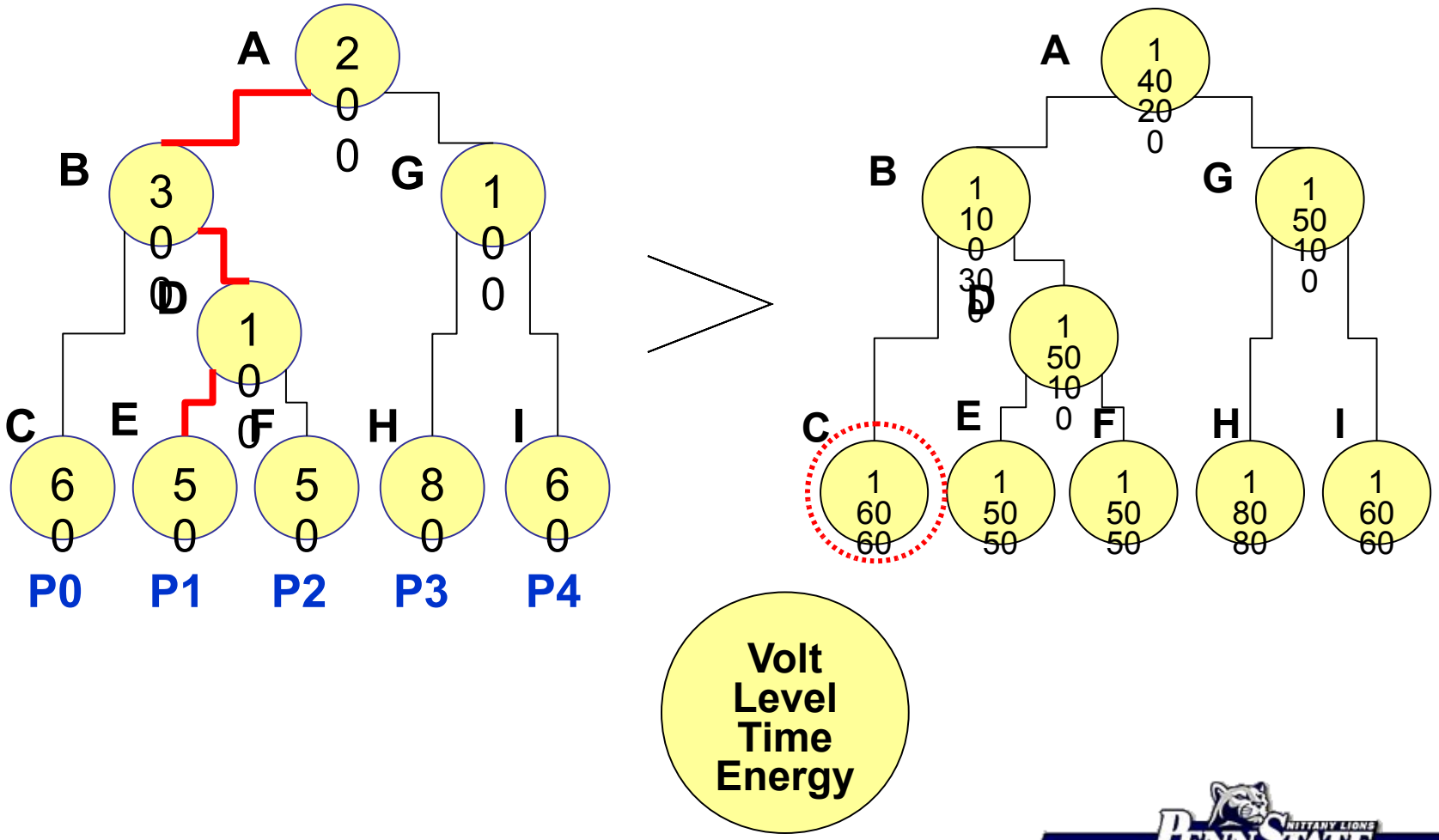


- Tree /hierarchical parallelism at core of many divide and conquer and multilevel methods

Example Weighted Tree



Example VTE Tree



Algorithm I

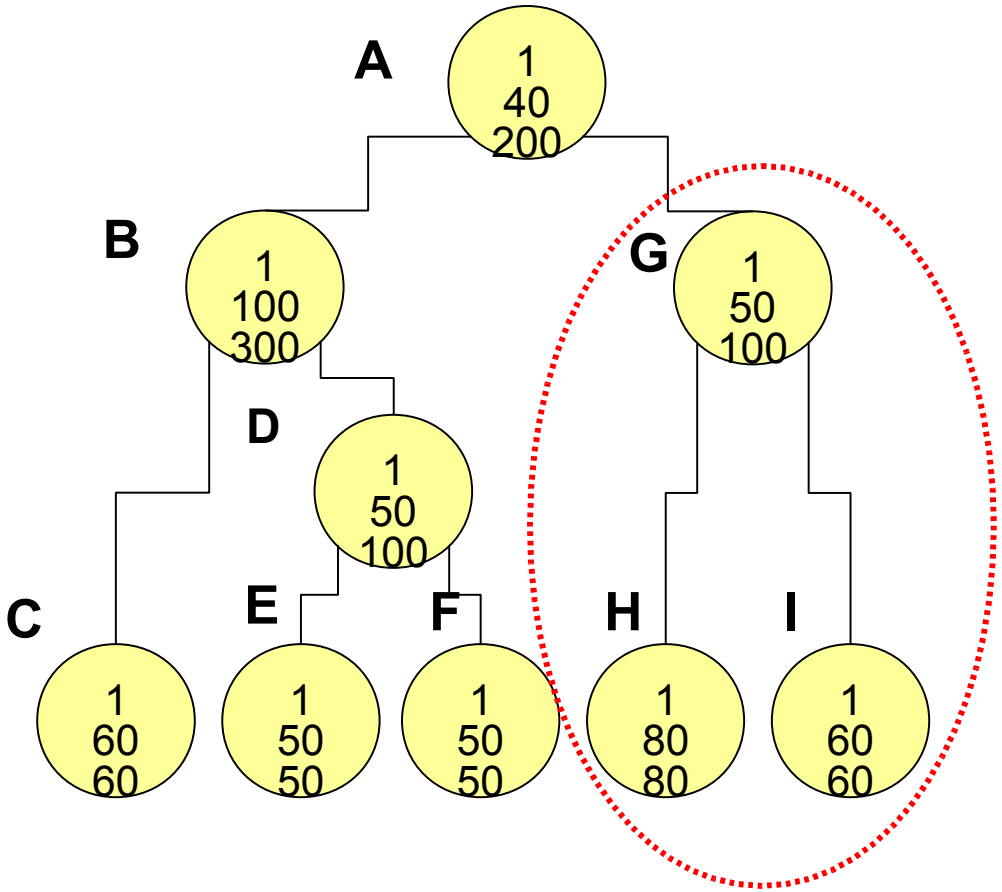
- At node x with children a and b
  - If W(a) > W(b), voltage scale T(b) proportional to W(b)/W(a) (using the nearest available V-F pair)
  - Use the scaling factor to update weights in VTE tree
- Start at the root and apply recursively
- Execution time is not increased
- Can be generalized to include scaling with time penalties

```
VoltageScale(node)
{
  Assign lowest possible V level to lighter subtree
  VoltageScale(leftSubtree)
  VoltageScale(rightSubtree)
}
```

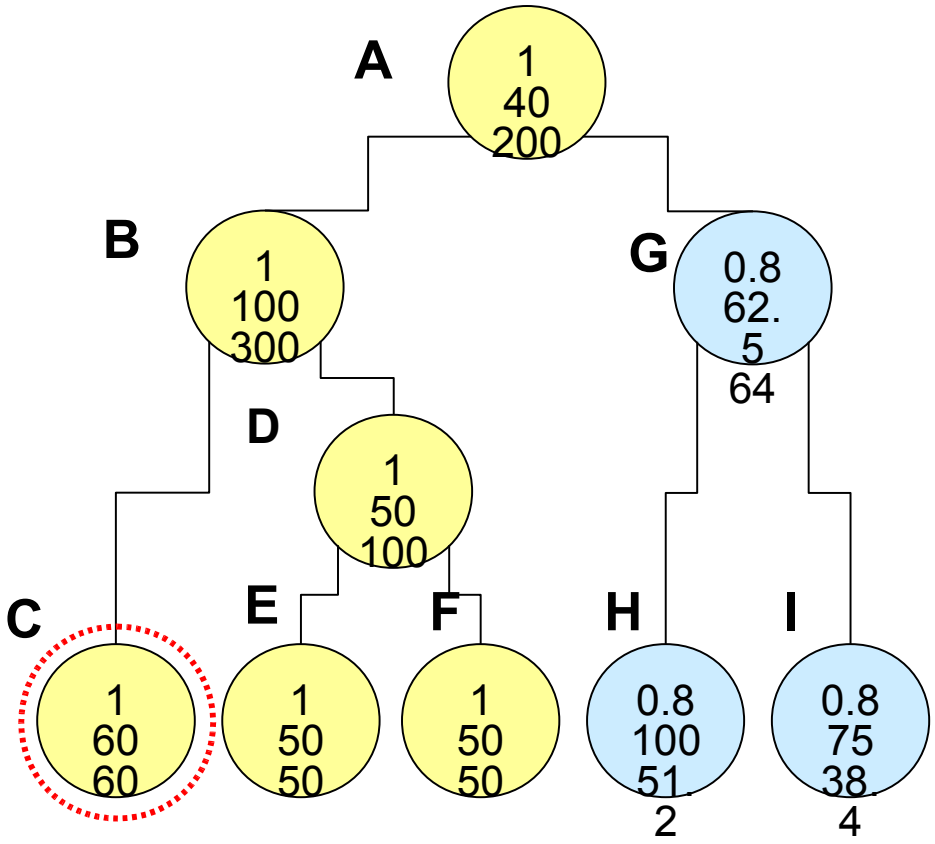
Volt/Freq/Power Levels in Examples

Voltage	Frequency	Power
1	1	1
0.8	0.8	0.512
0.6	0.6	0.216
0.4	0.4	0.064

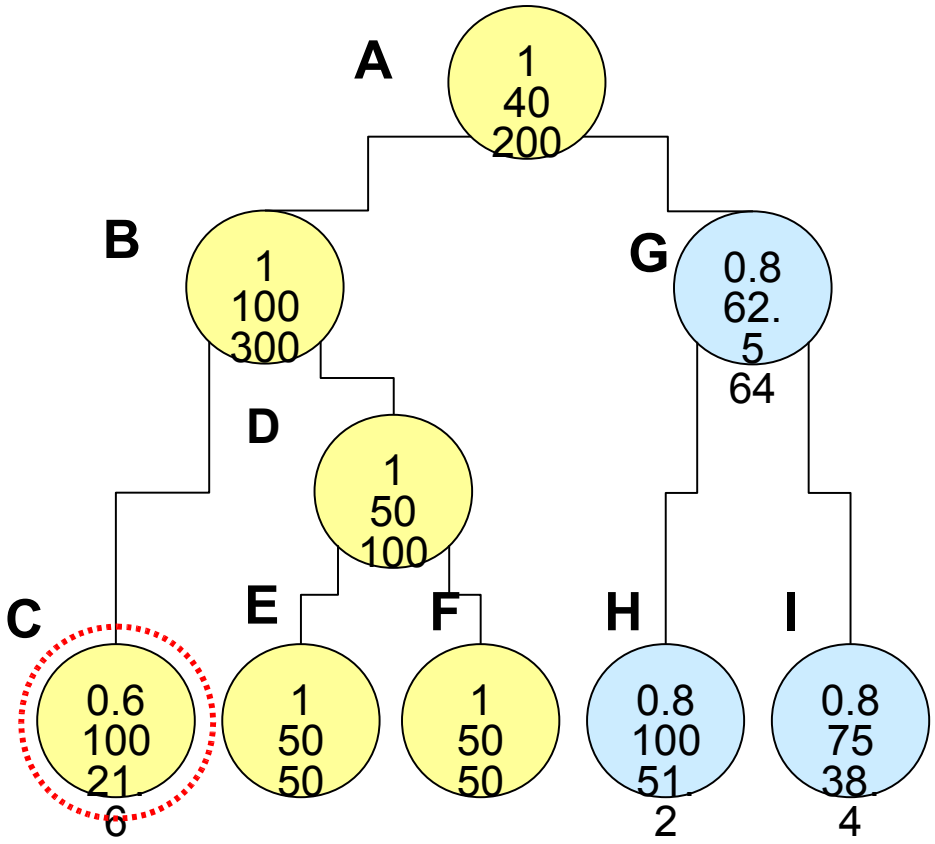
Algorithm I: Example



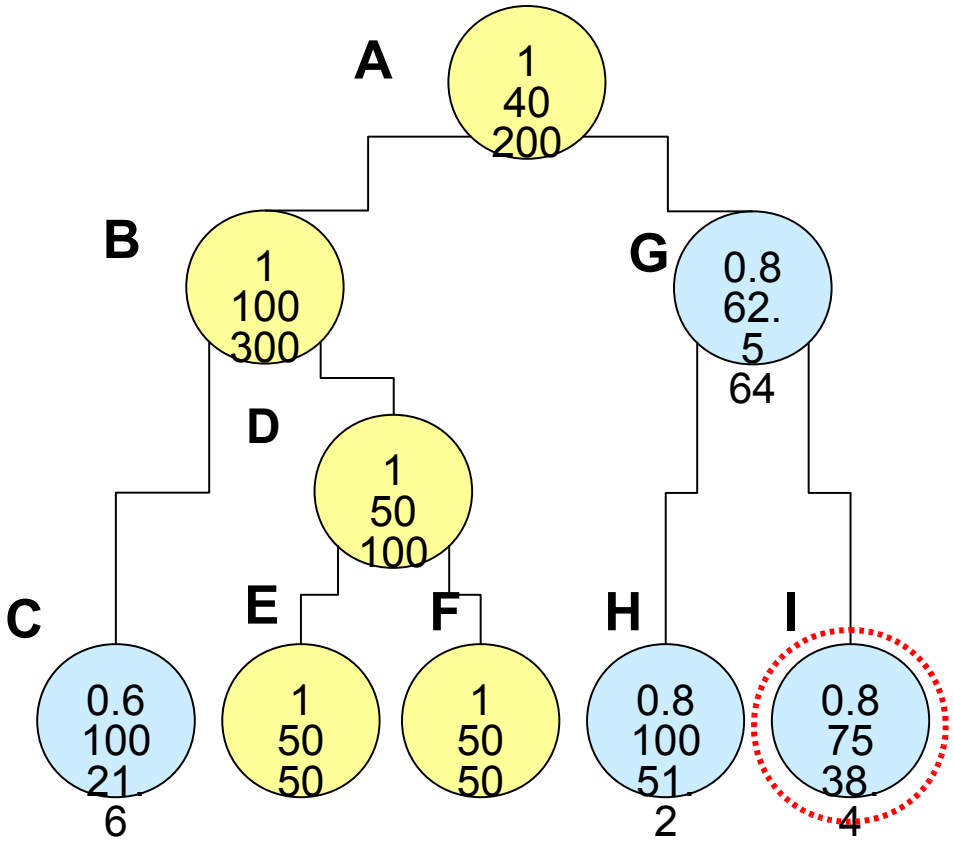
Algorithm I: Example



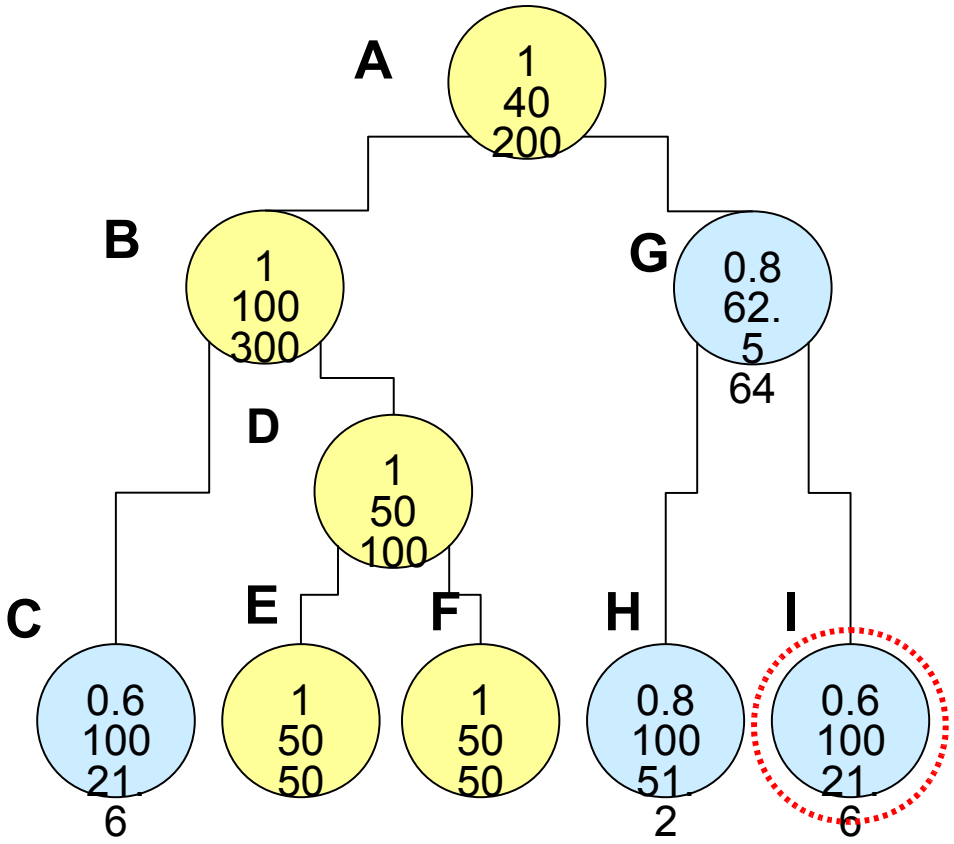
Algorithm I: Example



Algorithm I: Example

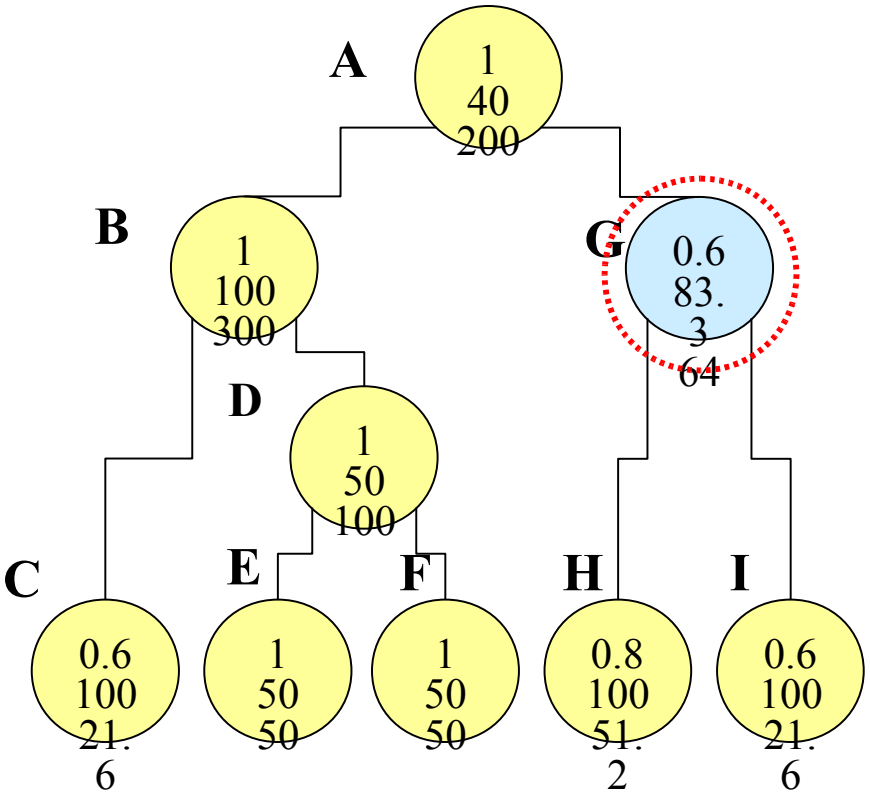


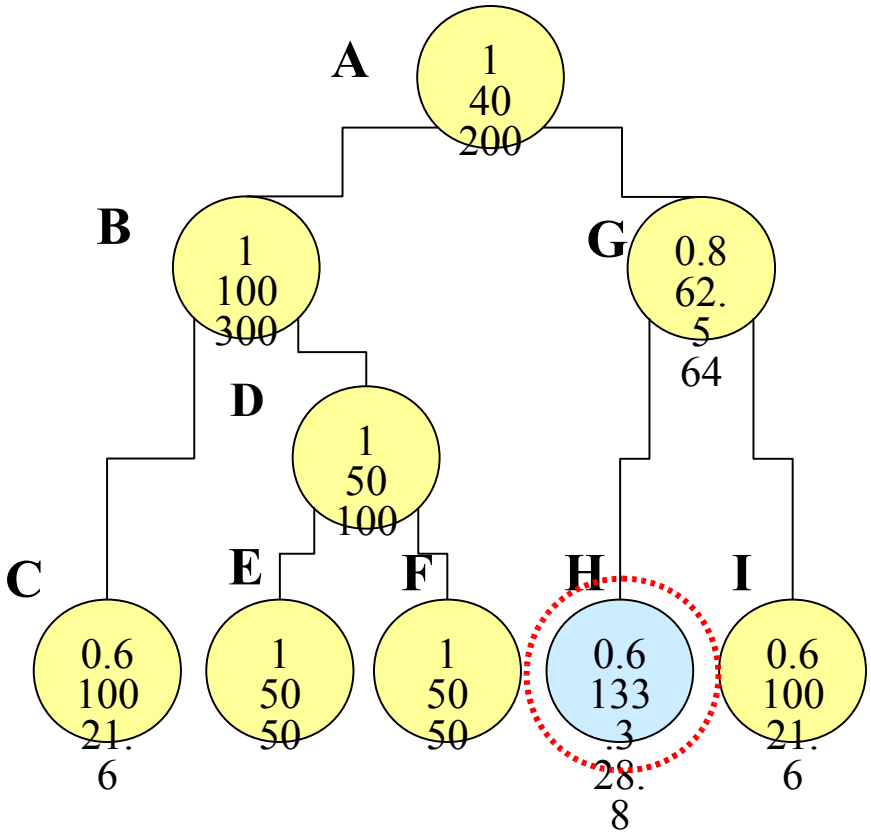
Algorithm I: Example



Energy consumption 858.4, a 14.2% reduction

Improving Algorithm I





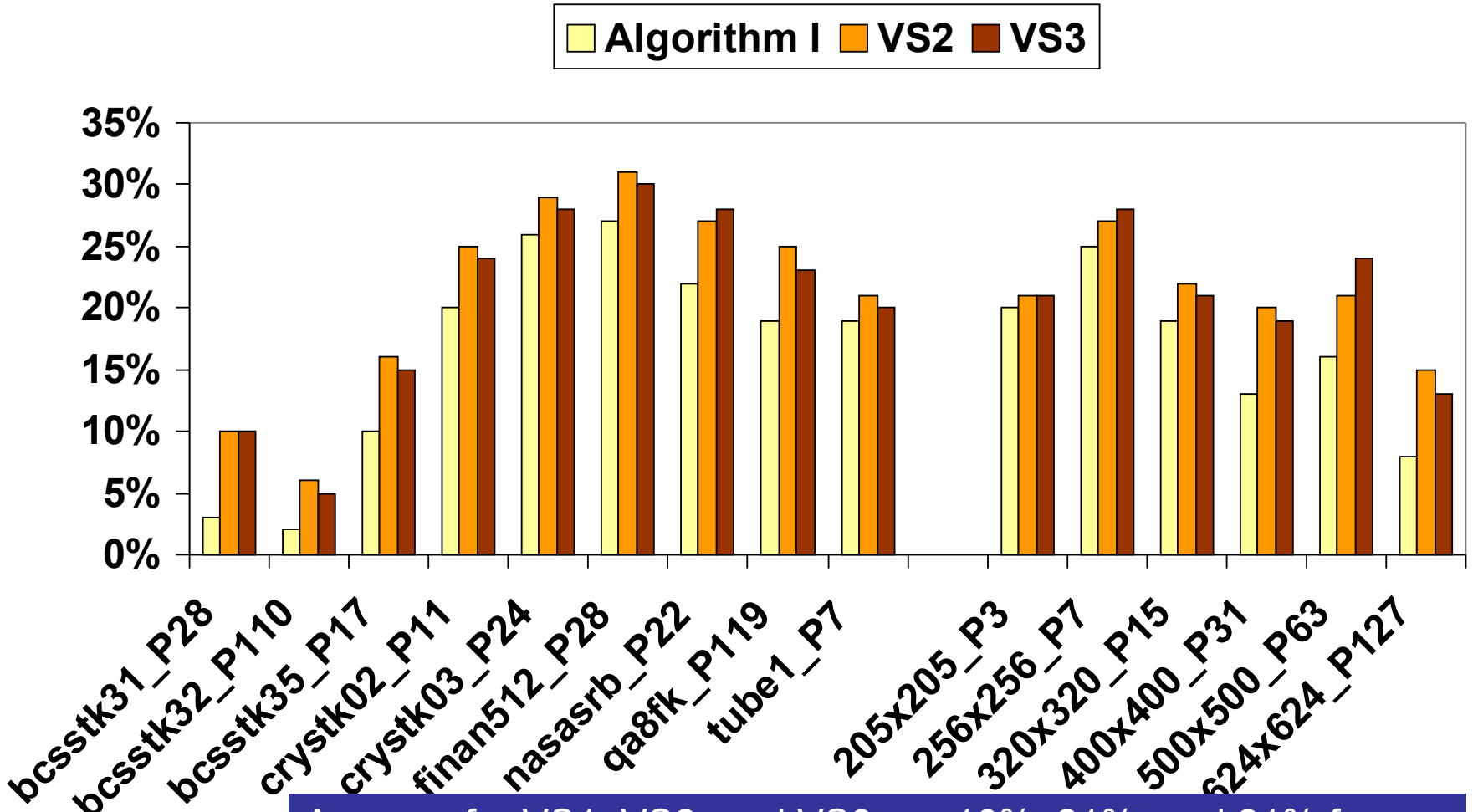
Refinements

1. Apply Algorithm I (VS1)
   - Refine by scaling individual nodes when feasible
   - Selection of nodes to scale
     - Root-first (VS2)
     - Child-first (VS3)
2. Allow performance degradation
   - *VoltageScale(node, 0)*: no degradation
   - P percent degradation: *VoltageScale(node, T*P)*
3. Consider CPU and link scaling
   - Use computation and communication loads

Experiments

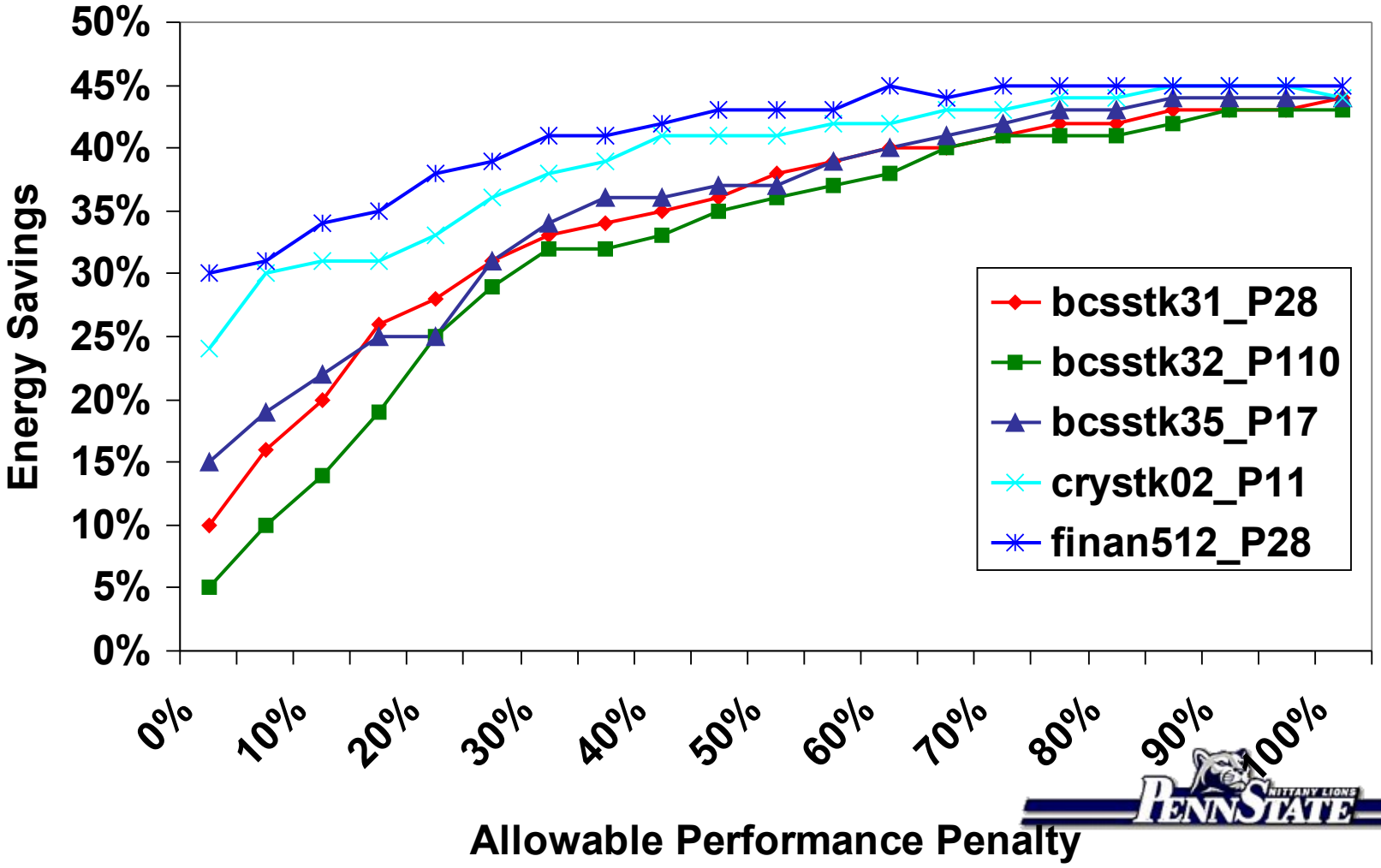
- Power numbers from Transmeta Crusoe
- 20 voltage levels
- Algorithms:
  - VS1:Algorithm I + individual node scaling *VS2*: Algorithm II (root-first) *VS3*: Algorithm II (child-first)
  - Allowing delays with VS3
  - Allowing CPU+Link Scaling

Energy Savings

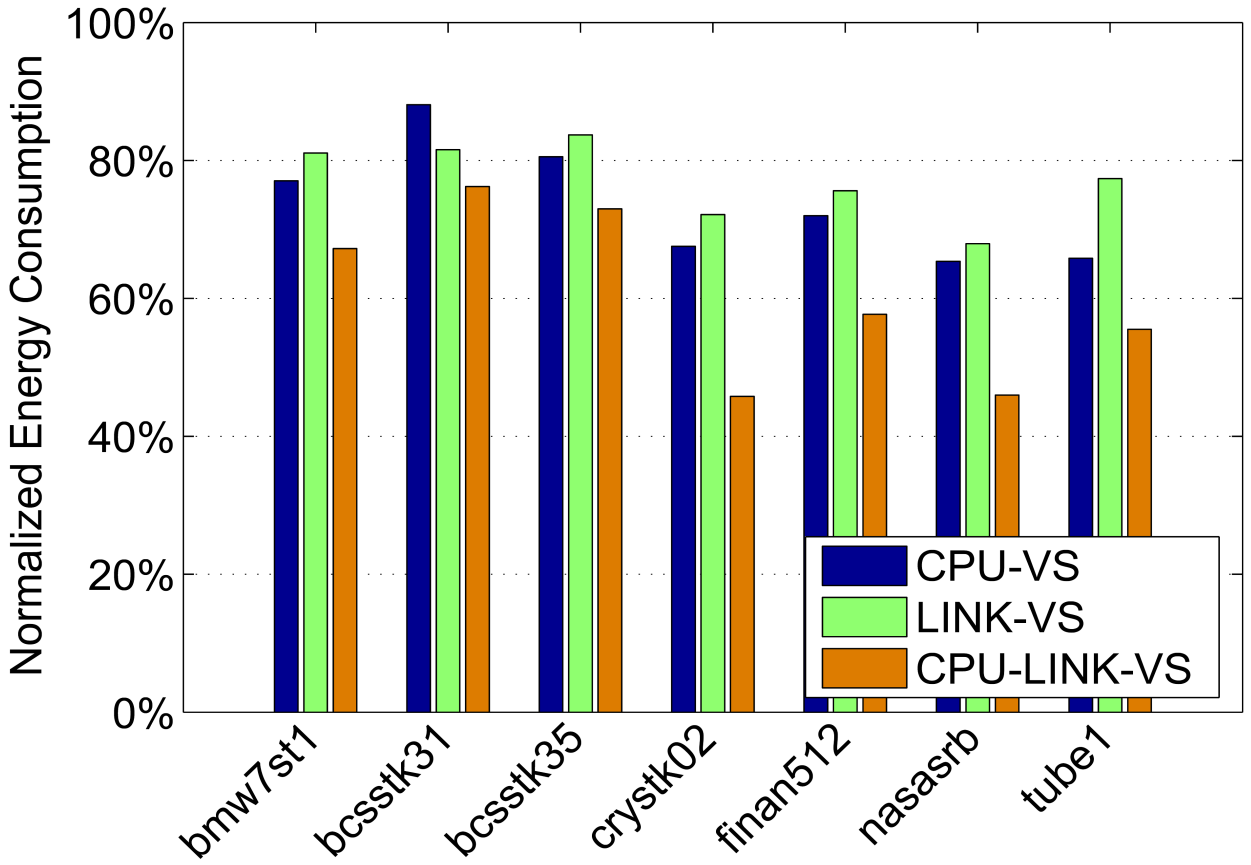


Average for VS1, VS2, and VS3 are 16%, 21%, and 21% for practical problems (17%, 21%, and 21% for model problems).

Allowing Delays (VS3)



CPU+Link Voltage Scaling



Average Savings: CPU-VS (27%), LINK-VS (23%), CPU-LINK-VS (40%)

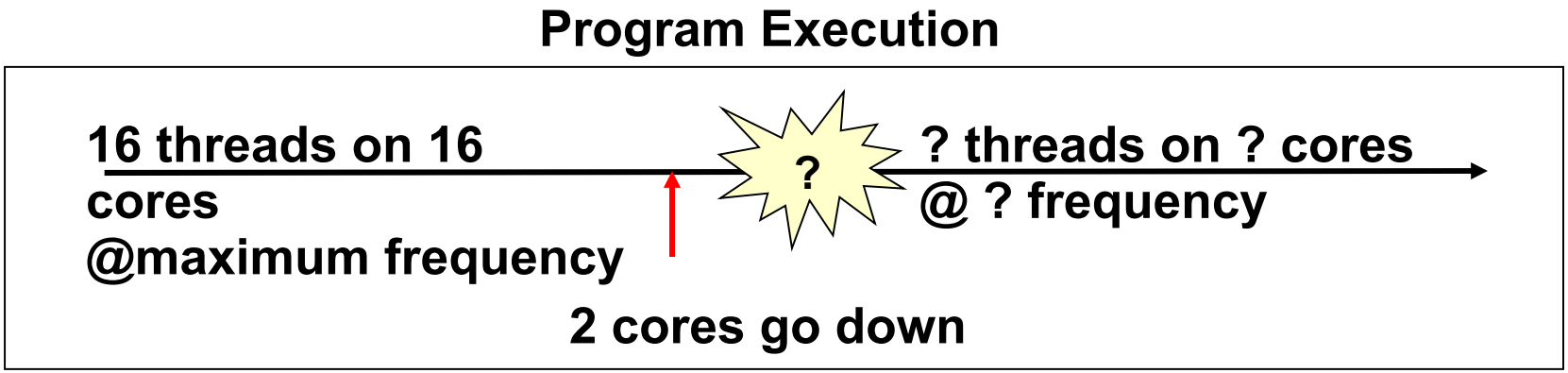
Tree-Workloads and V-F Scaling

- Indicates potential for transforming imbalances to energy savings
- M. Sato et. al : V-F saving on transmeta cluster on general workloads and corroborated our results
  - Focused on real V-F scaling models taking into account
    - overheads for scaling up/down
    - link sharing

II: Energy-Aware Algorithms

- II (c) Opportunities at a multicore node
  - Scheduling for energy and reliability
  - IPDPS-08, Yang Ding, Mahmut Kandemir, P. R

Energy-Aware Adaptation to Failures



Scenarios

1. Change number of cores
2. Change number of cores and number of threads
3. Change number of cores, number of threads, and voltage/frequency levels

Mechanism

Helper thread + dynamic scheduling

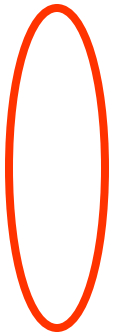
Function-based adaptivity

Problem

- Can we adapt application execution to changes in core availability or core loads for enhanced energy & performance efficiency +
- Reduce energy delay product (EDP) at runtime by varying
  - Number of cores
  - Number of threads
  - Voltage/frequency levels

Number of Threads Matters

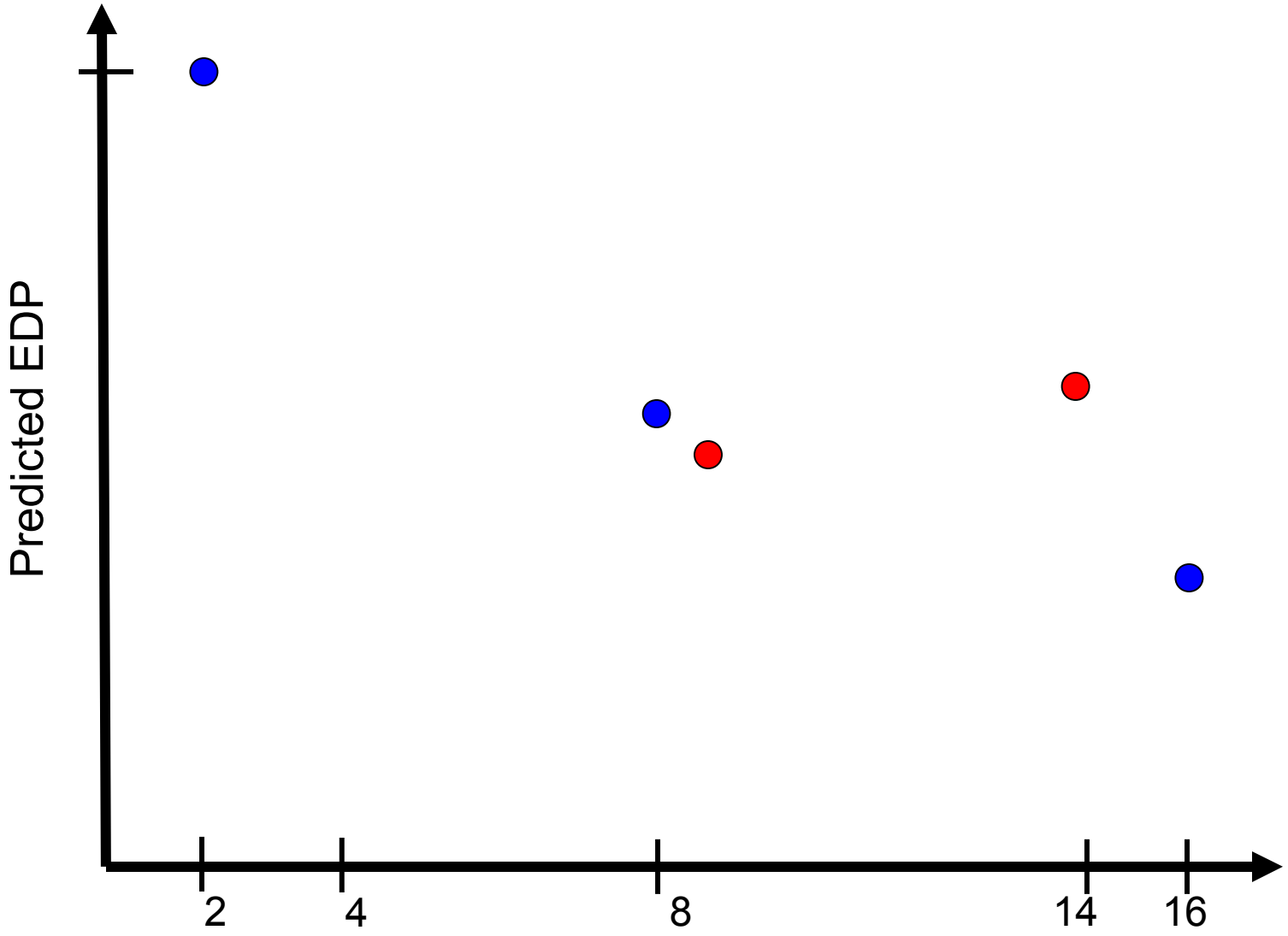
16 threads: 0.267
8 threads: 0.218
Improvement: 18%



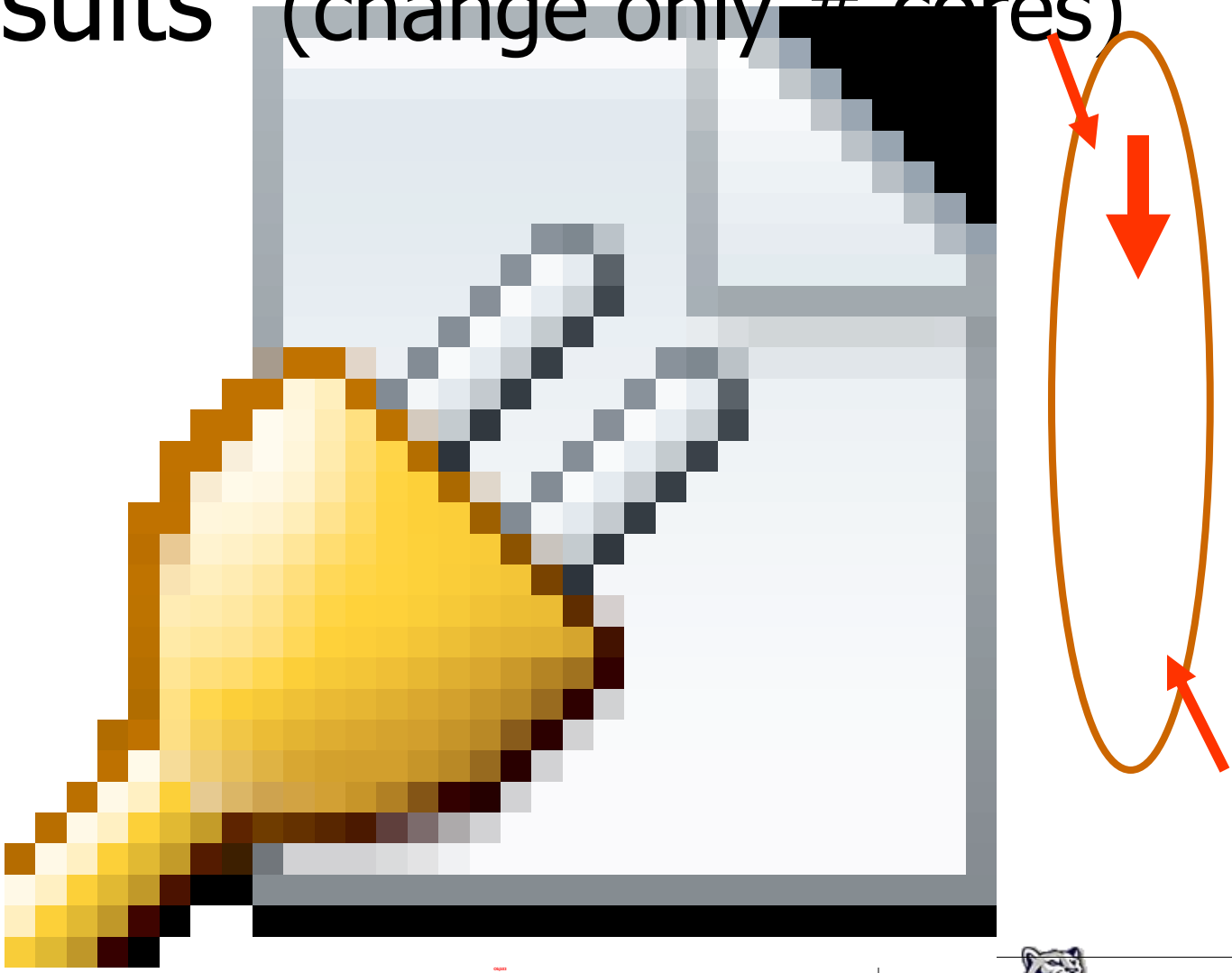
Helper Thread Based Adaptation

- Helper thread's recipe
  - Collect performance/energy statistics as the execution progresses
  - Calculate EDP values
  - Use data fitting with available data and interpolate to predict configuration
  - Will work EDP on parameter space is reasonably smooth

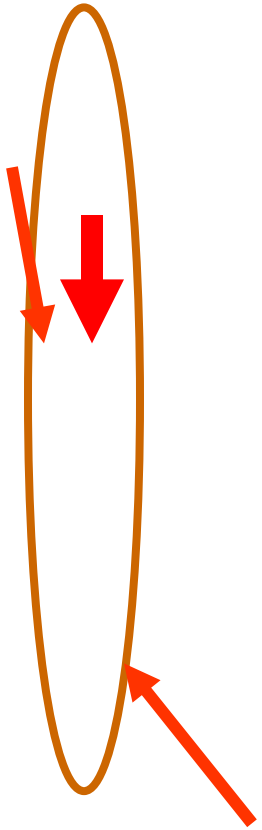
Data Fit +Interpolate (change only # cores)



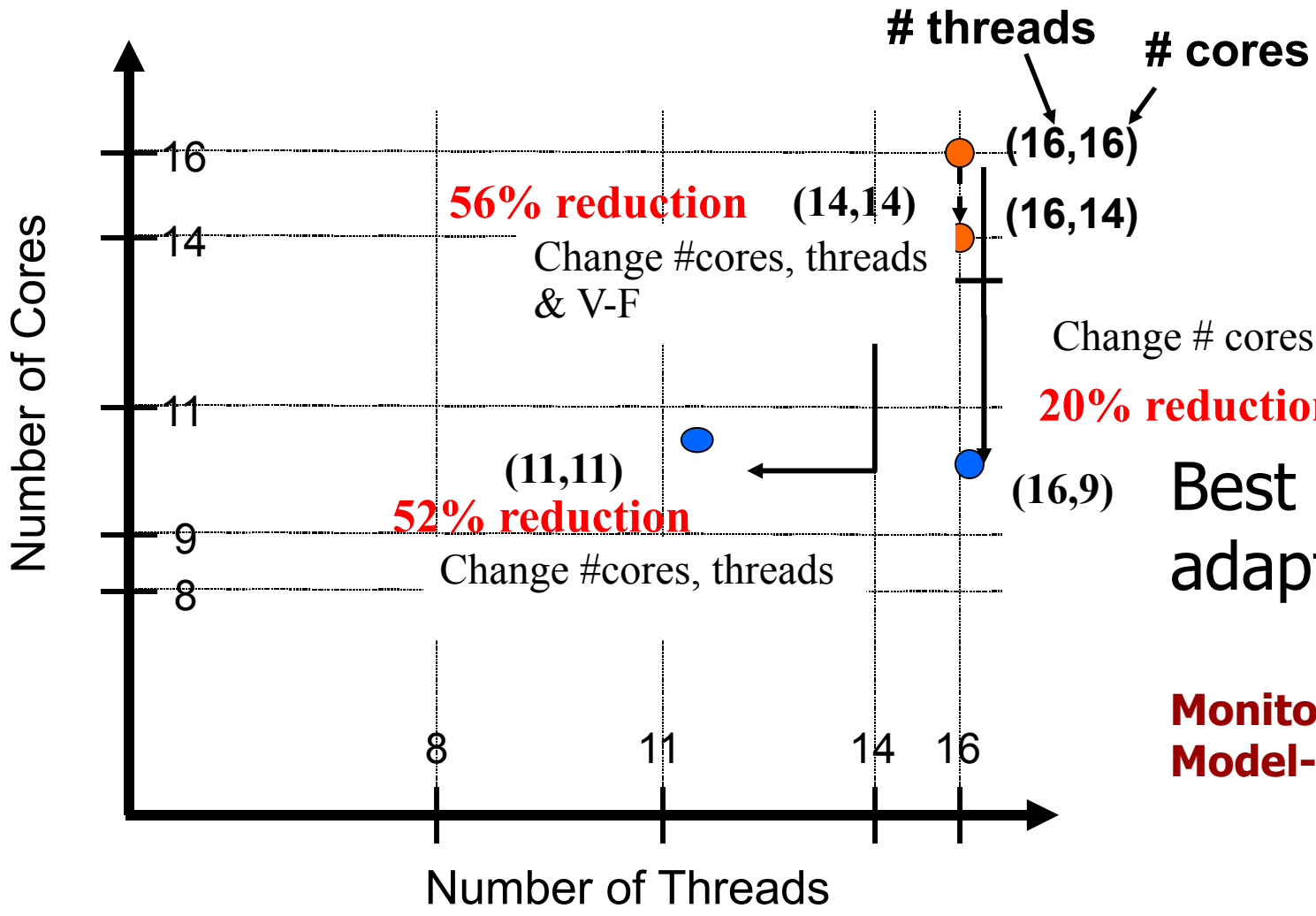
Results (change only # cores)



Results (change # cores & # threads)



EDP Landscape for Multigrid



[Ding, Kandemir, Raghavan, Irwin, IPDPS'08]

Summary of Results

Summary

- Algorithm/software redesign needed for sparse/irregular codes to
  - Convert performance gaps into energy savings
  - Schedule for higher performance at lower energy
- Across nodes in large nets: Control network energy dynamically: opportunities even in a collective communication
- Across nodes: Convert load imbalance to energy savings
- At node: Schedule for performance, energy when there are h/w or load variations

- Software control of data-staging

Acknowledgements

- Joint work with:
  - Gulin Chen, Sarah Conner, Yang Ding and Konrad Malkowski
  - Mary Jane Irwin and Mahmut Kandemir
- Support from:
  - NSF, DoD, IBM
  - Institute for CyberScience@PSU
- Thanks to:
  - Google Images, Wikipedia, ARSTechnica
  - Scheduling 2009 Organizers, audience

PennState