Optimizing Scientific Software

Padma Raghavan

The Pennsylvania State University

Research Supported by NSF, DoD & IBM

Collaborators: B. Cover, Y. Ding, K. Malkowski & M. Kandemir

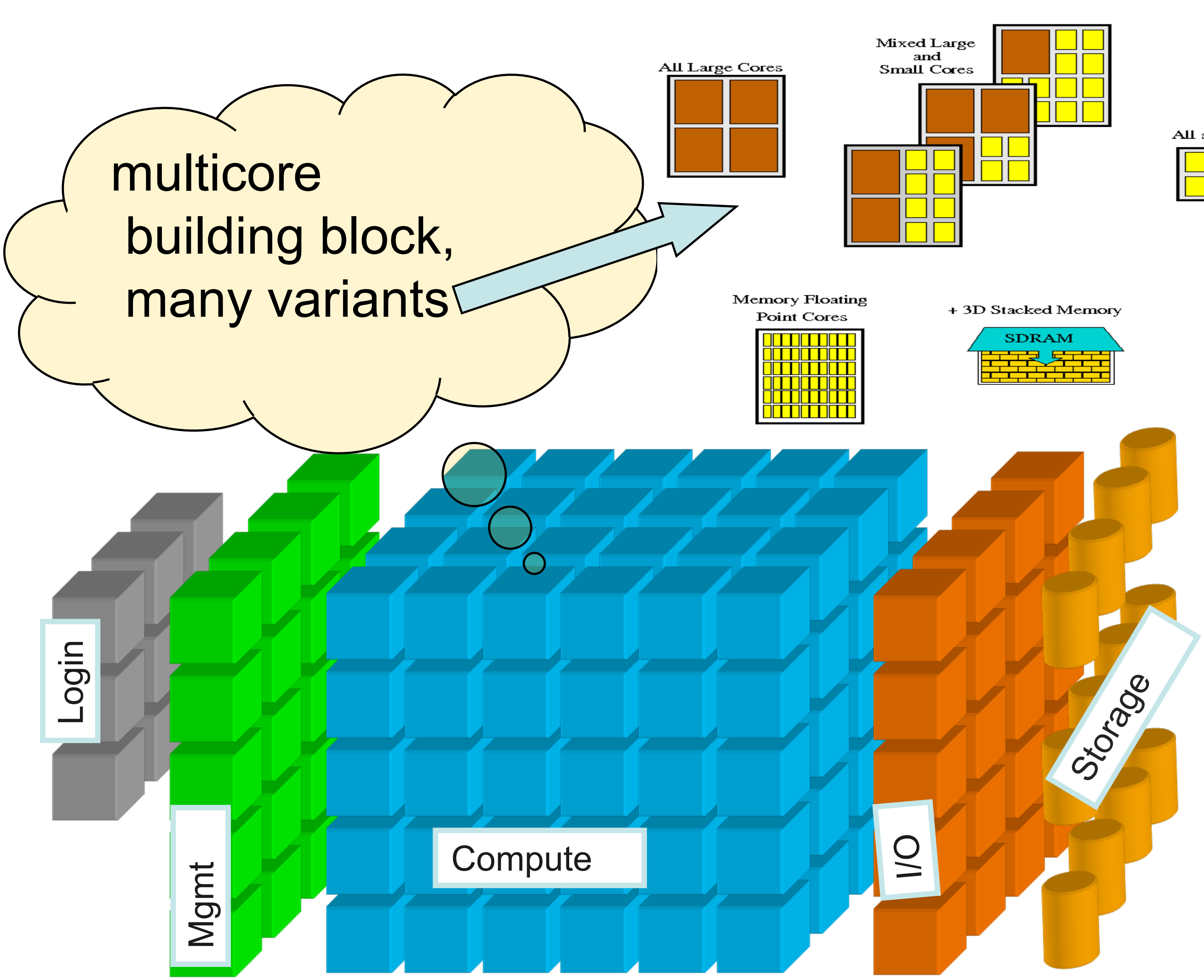
Clusters, Clouds, and Grids for Scientific Computing, 2010

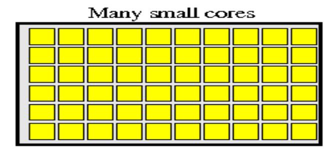
This Talk

- The multicore node at exascale
  - Impacts on algorithm & s/w design
- Scaling & efficiency challenges@multicore
  - Examples of h/w-s/w co-adaptations for performance, **energy** & soft error resiliency
- Opportunities for h/w-s/w co-evolution

I: Exascale Architectures

- Then, now and beyond
  - From fast, hot … to
  - Billion-way parallel, hetErogeneous, cooler, unreliable

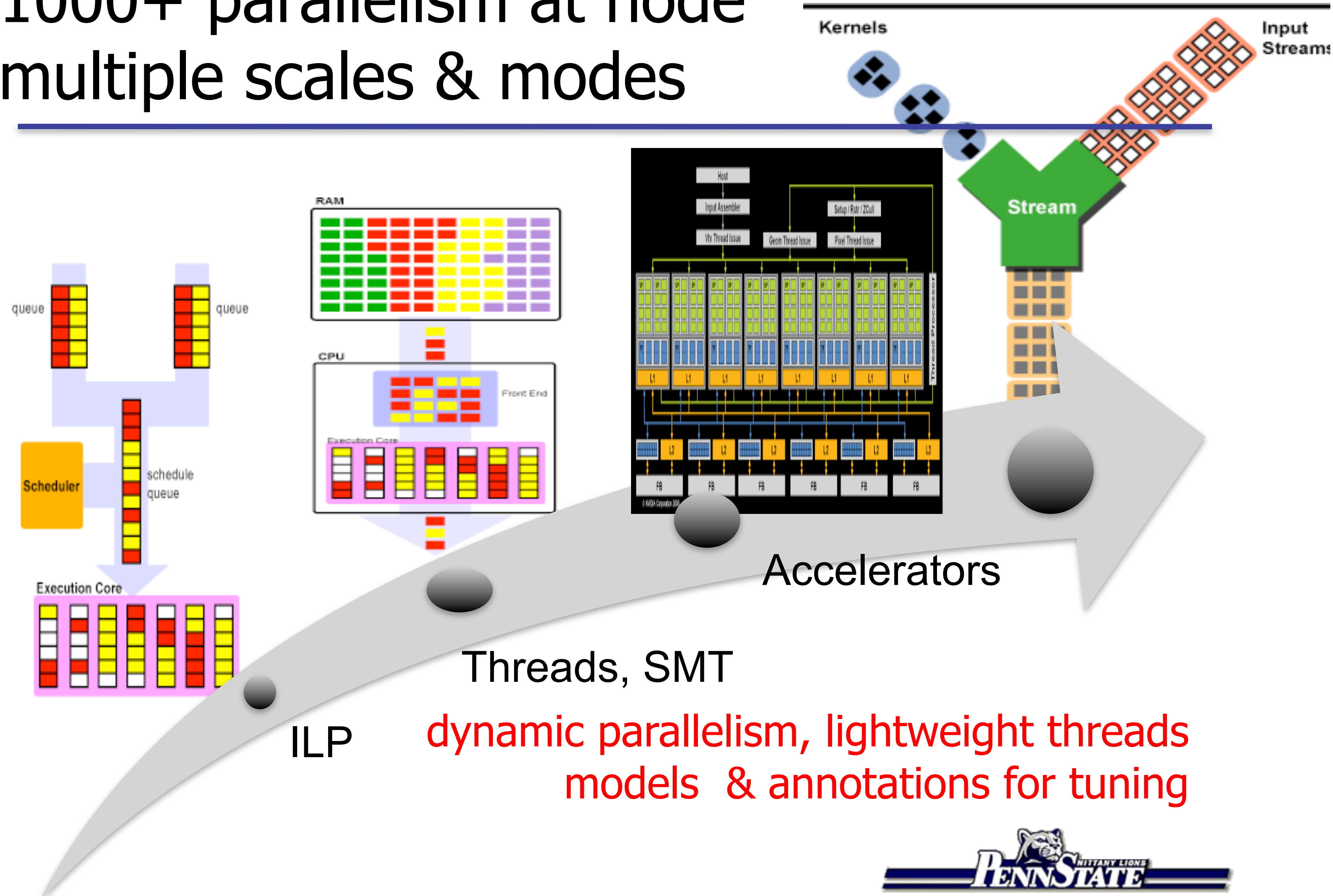




- ✓Expect rapid changes@ higher core counts
- ✓Evolving design of on chip networks, caches & sharing

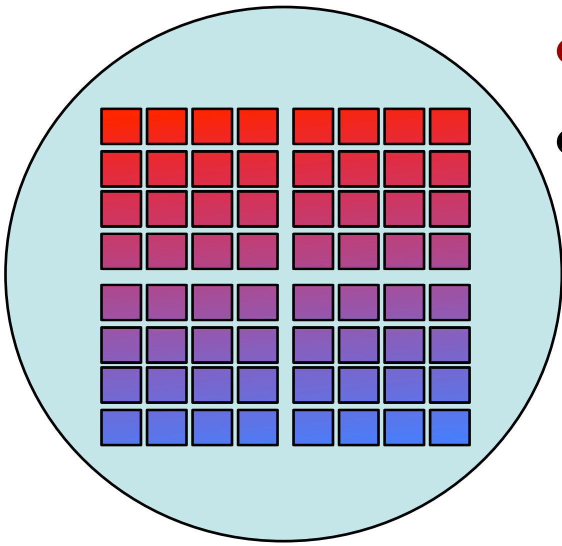
From Rajeev Thakur's talk@CCGSC'2010

1000+ parallelism at node
multiple scales & modes

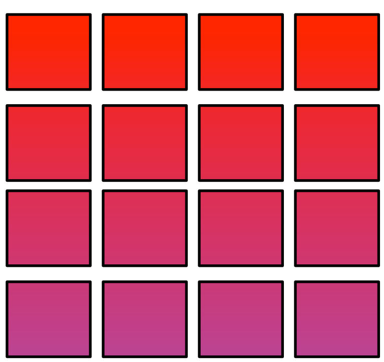


dynamic parallelism, lightweight threads
models & annotations for tuning

Process Variability



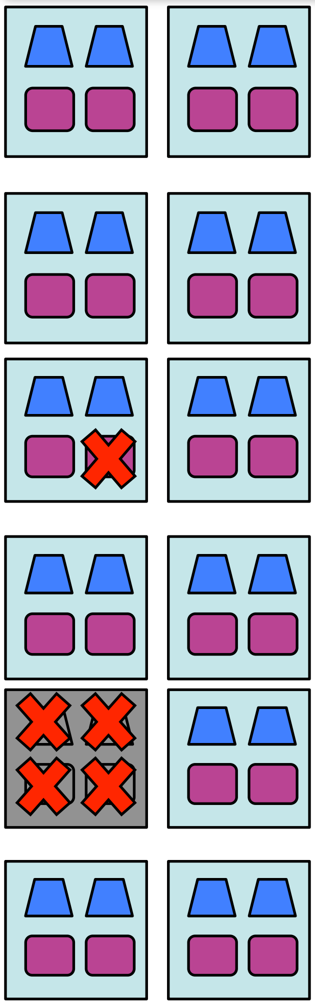
- Manufacturing is imperfect
- Die for 4 chips@ 16 cores
    - Top fast, high leak
    - Bottom slow, low leak
    - Variations within chip



- Algorithms/software redesign for variations

- Even dense regular codes will have irregular DAGs

Failures & Soft-Errors



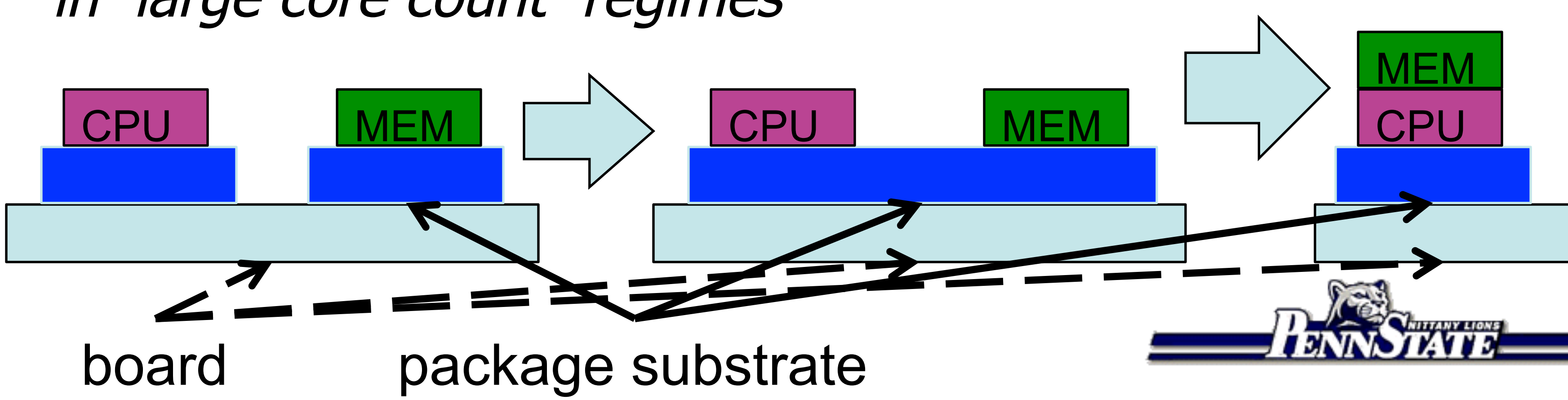
- Components will fail; use cores in diminished capacity
- Soft errors (bit flips) in low V regimes impact algorithm correctness
- Algorithms/software redesign for resiliency
- *Even dense regular codes could lose structure when redesigned for resiliency, e.g., through selective protection based in numeric attributes*

Caches & Memory

- Caches (esp. large shared L2//L3) not useful for applications w/o reuse or long reuse distances

- Power efficient options such as user programmable memories

- Needs Algorithm/software redesign
- *Even dense codes may not find large L2/L3 cache useful in large core count regimes*

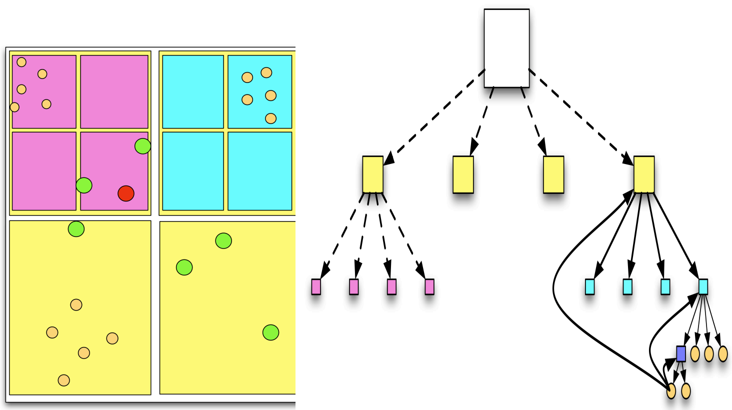


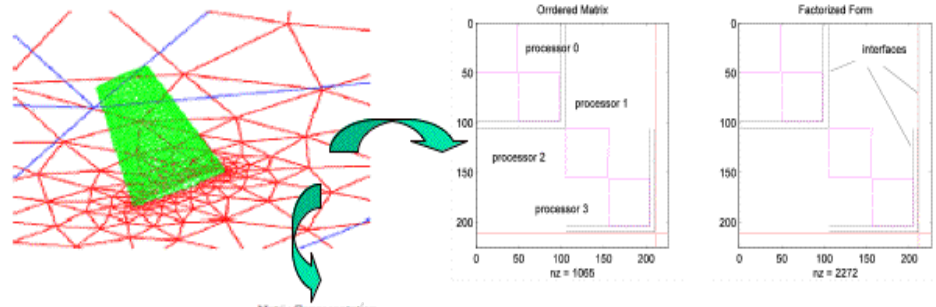
Sparse/ Irregular Computations

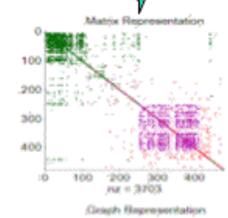
Interoperable, Sparse Data Structures and Transformations

Examples of tradeoff space for optimizations

Discretizing with adaptive meshes

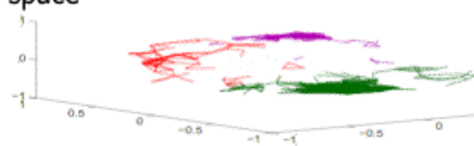




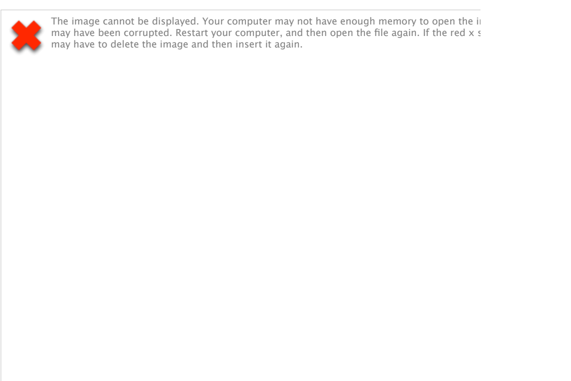


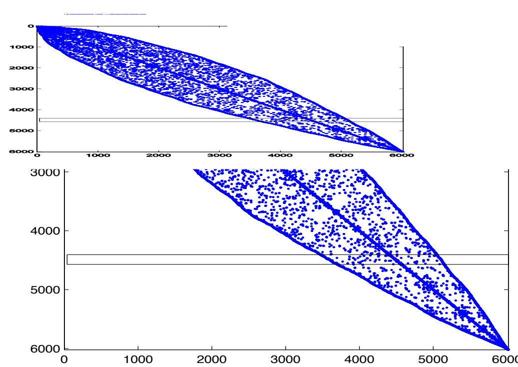
Sparse matrices, automatically ordered to reveal hierarchical structure

Graphs, networks, clusters in low dimensional space



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

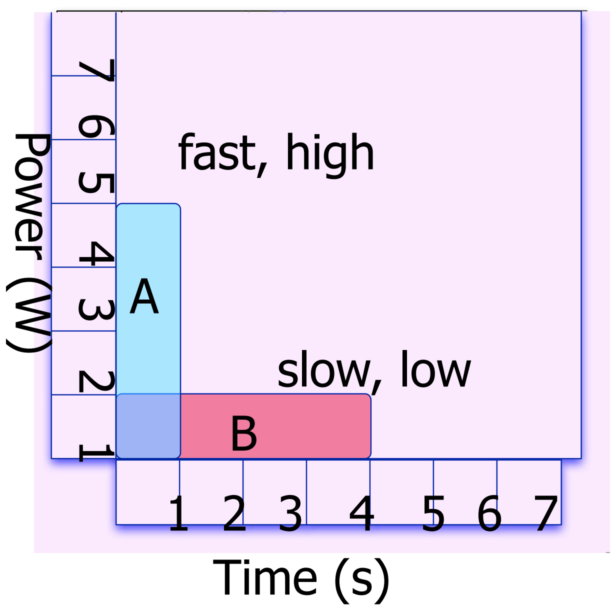


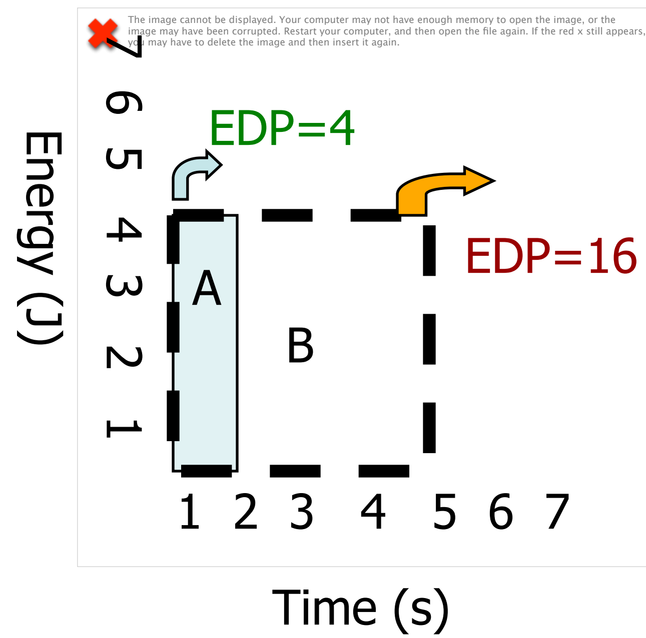


Application requirements-> algorithm selection + tuning -> H/W, S/W adaptivity

Measuring Energy Efficiency

Same code on two different systems A and B

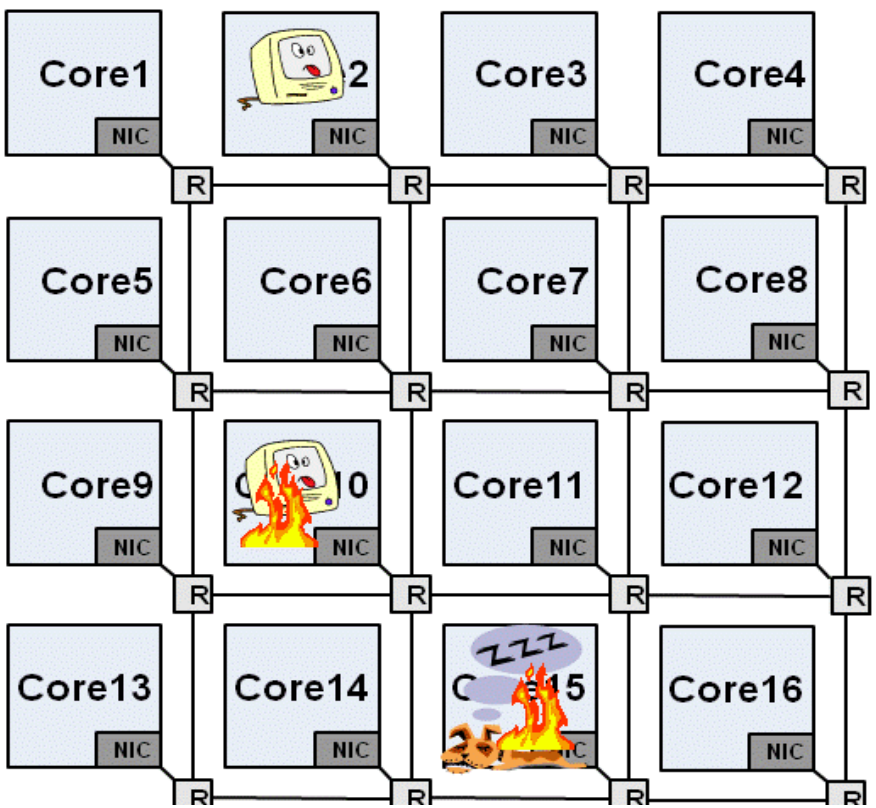




Equal energy (PDP) does not differentiate A from B

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

1. Schedule & Adapt to Failures

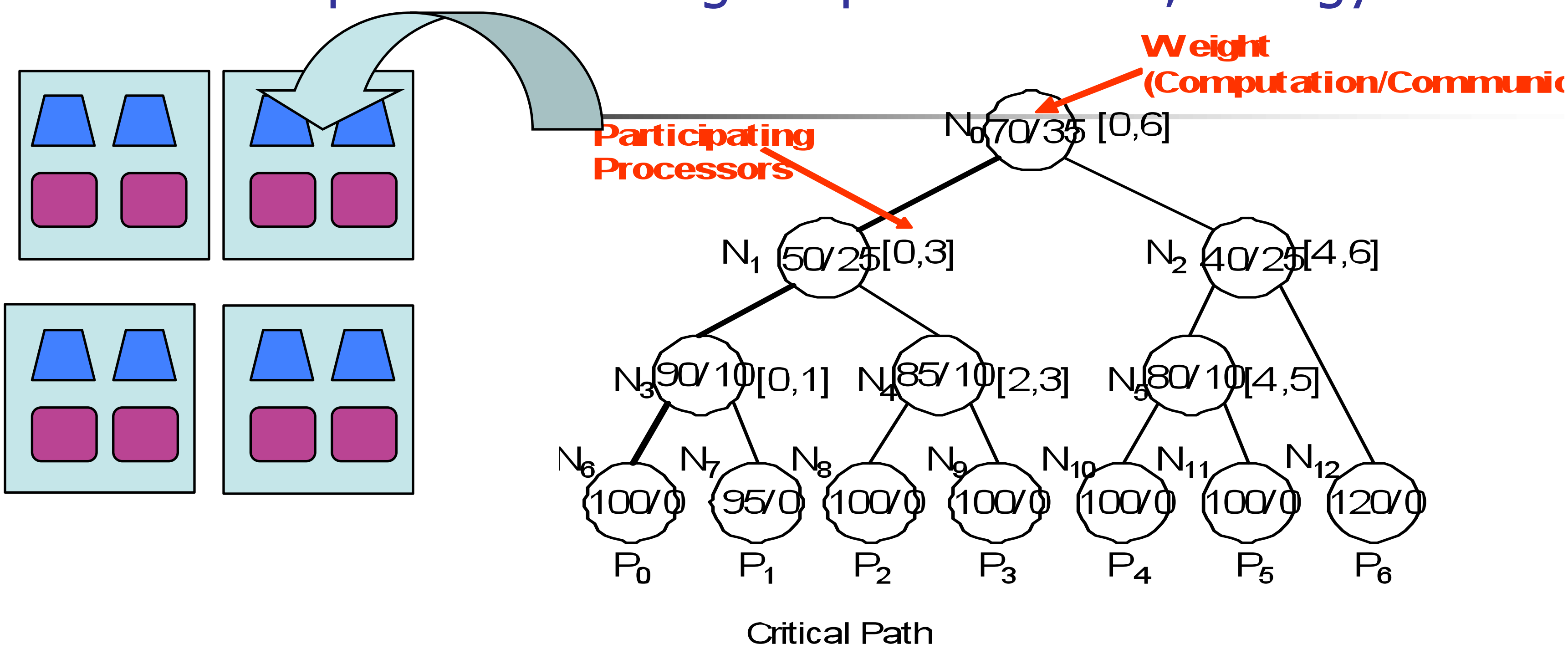


- Resiliency issues
- Component failure
- Thermal emergencies

Ding et al., JPDC 2009

Step 1.

- Consider fixed problem energy & performance efficiency at a multicore node
- Model core variations
- Critical path scheduling for performance, energy



Step 2. Adapt to Failures

Program Execution

16 threads on 16 cores @maximum frequency → **?** → **? threads on ? cores @ ? frequency**

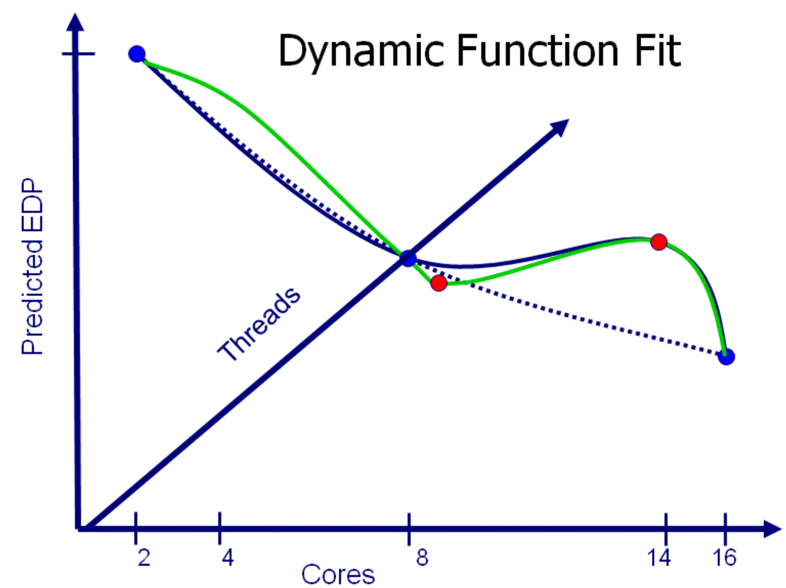
2 cores go down

Scenarios

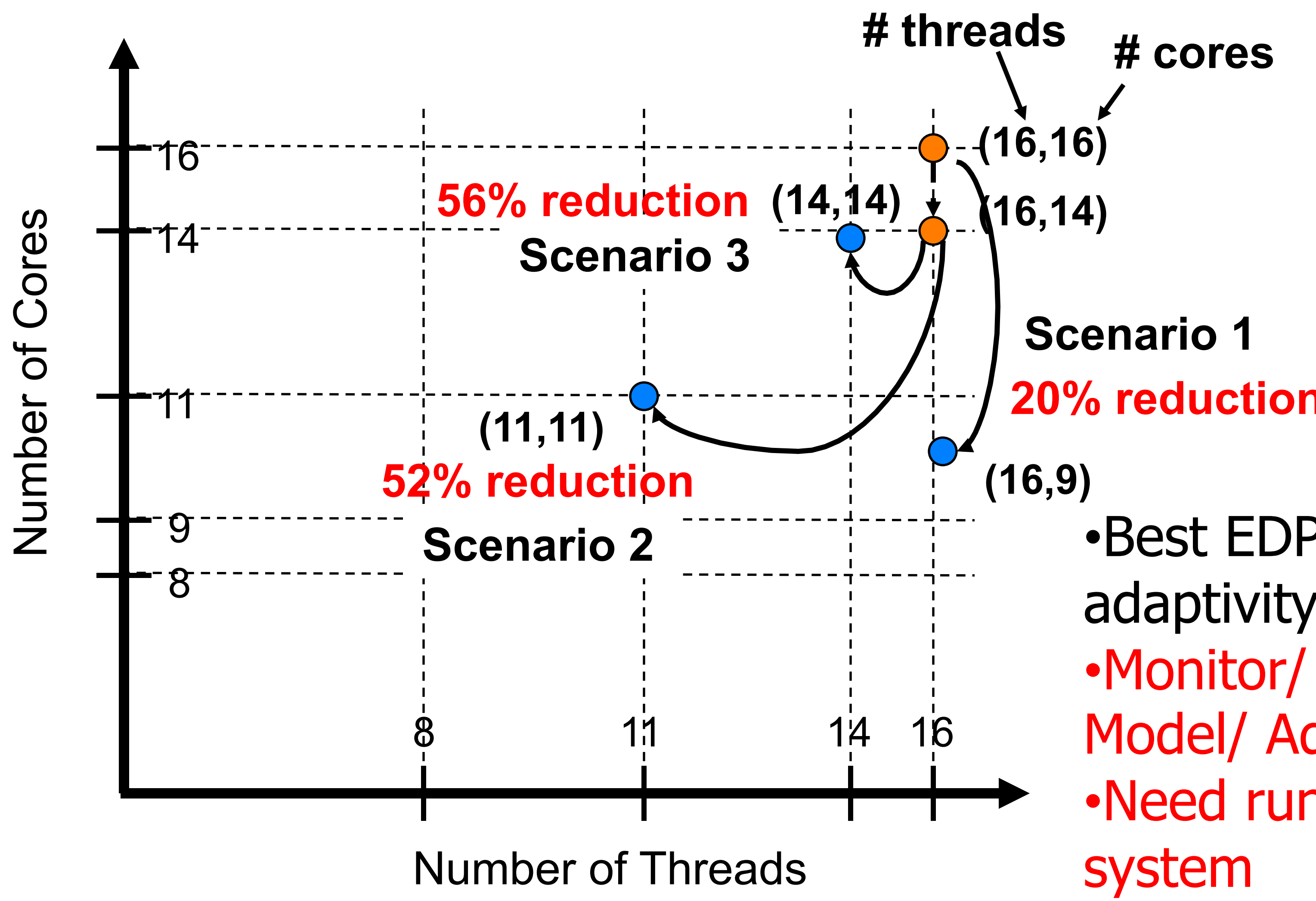
- Change number of cores
- Change number of cores & threads
- Change number of cores, threads, and voltage/frequency levels

Mechanism

- Function-based adaptivity
- Helper thread monitors, models, migrates



EDP Landscape for Multigrid

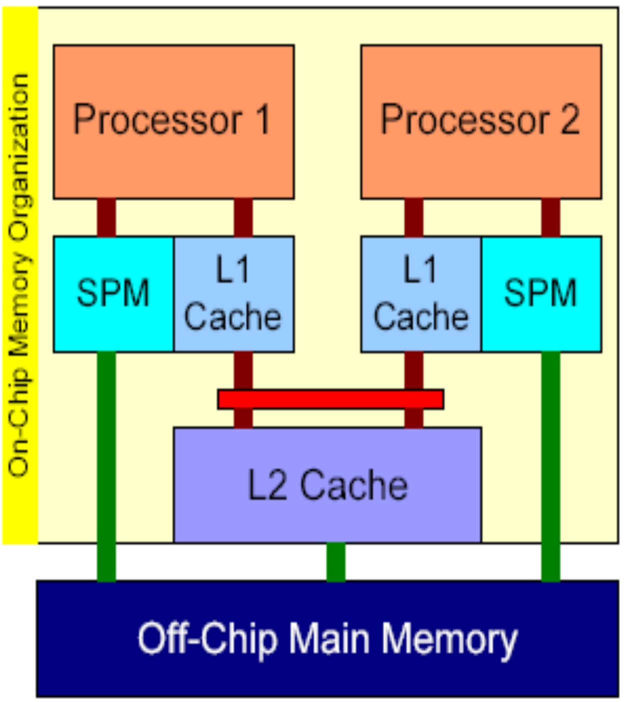


- Best EDP adaptivity?
- Monitor/ Model/ Adapt
- Need runtime system

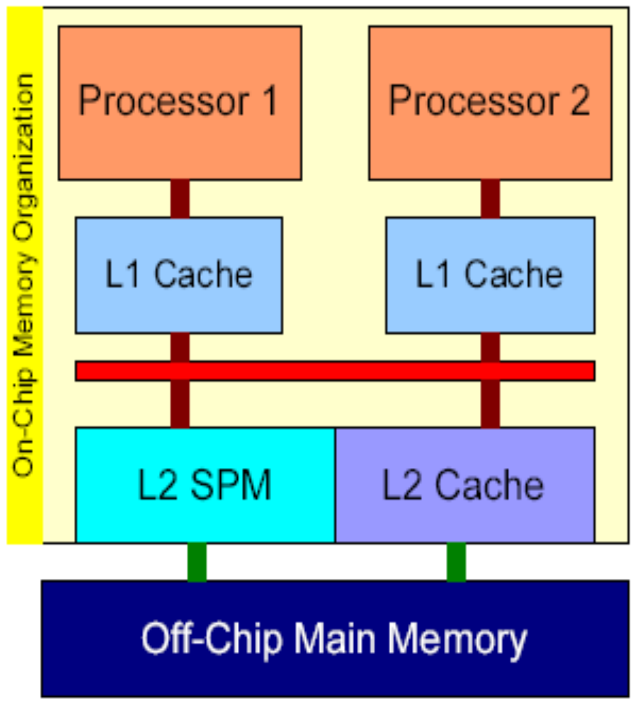
[Ding et al, IPDPS'08]

2. Caches x SPM Tradeoffs

SPM in L1



SPM in L2, Split



SPM in…..

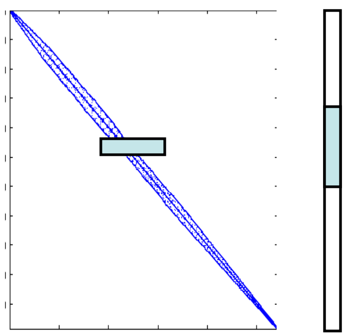
What is the effect of SPM at all levels (fixed number of bits per hierarchy level)?

Cover et al. IPDPS 2008

Power x Performance Efficiency

- Data & Power Locality

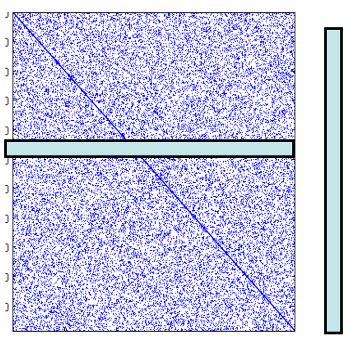
y←Ax



RCM

Data when and where it can be computed upon (data locality)

Power when and where it enables useful activity (power locality)



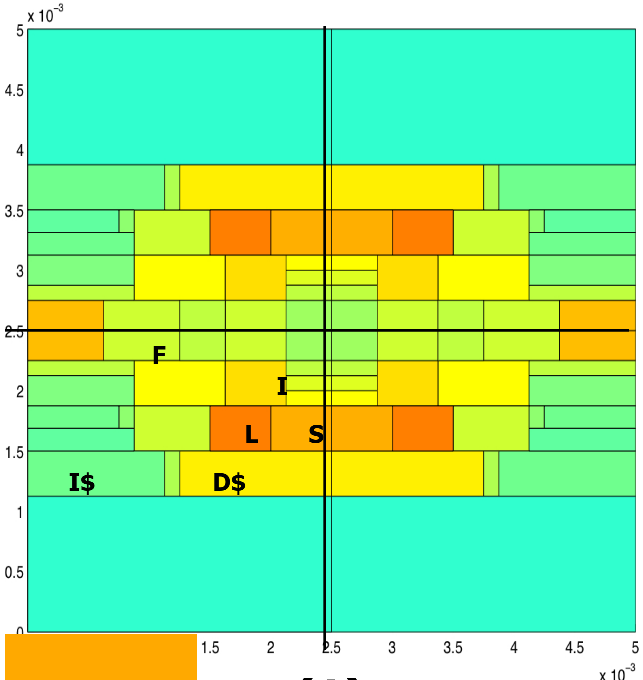
RND: Random

- Efficiency: Fraction relative to DGEMM for sparse matrix vector multiplication (SMV)
- SMV varietals:
  - CSR format: RCM, RND
  - Ordering X Blocking to increase locality in x … Toledo, PR, ….. Vuduc, Yelick..

Temperature Evolution (4-core)

DGEMM, SMV_RCM, SMV_RND

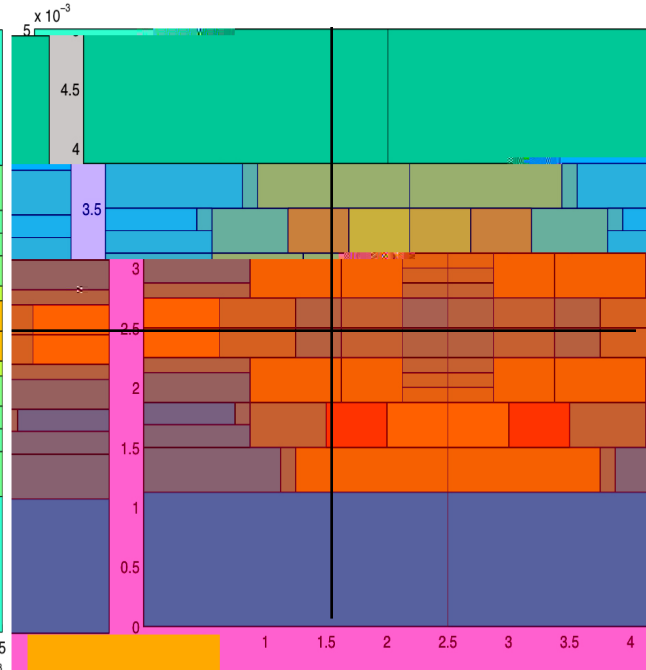
??



Video Clip

(1)

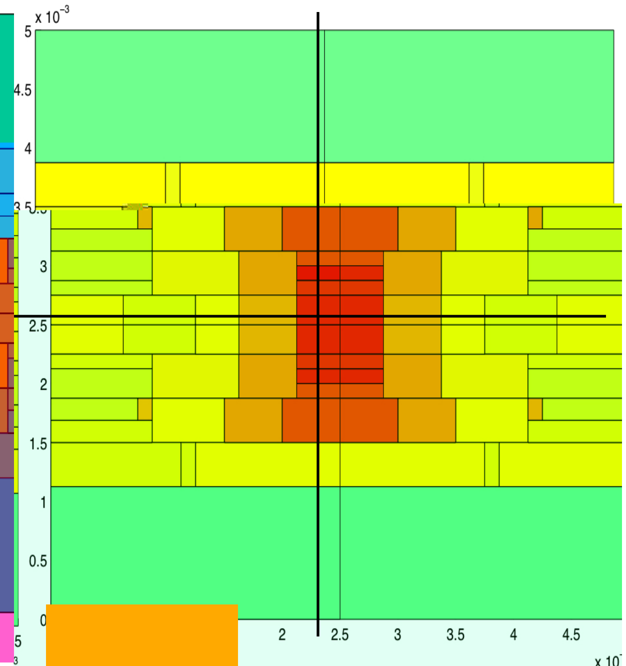
??



Video Clip

(2)

??



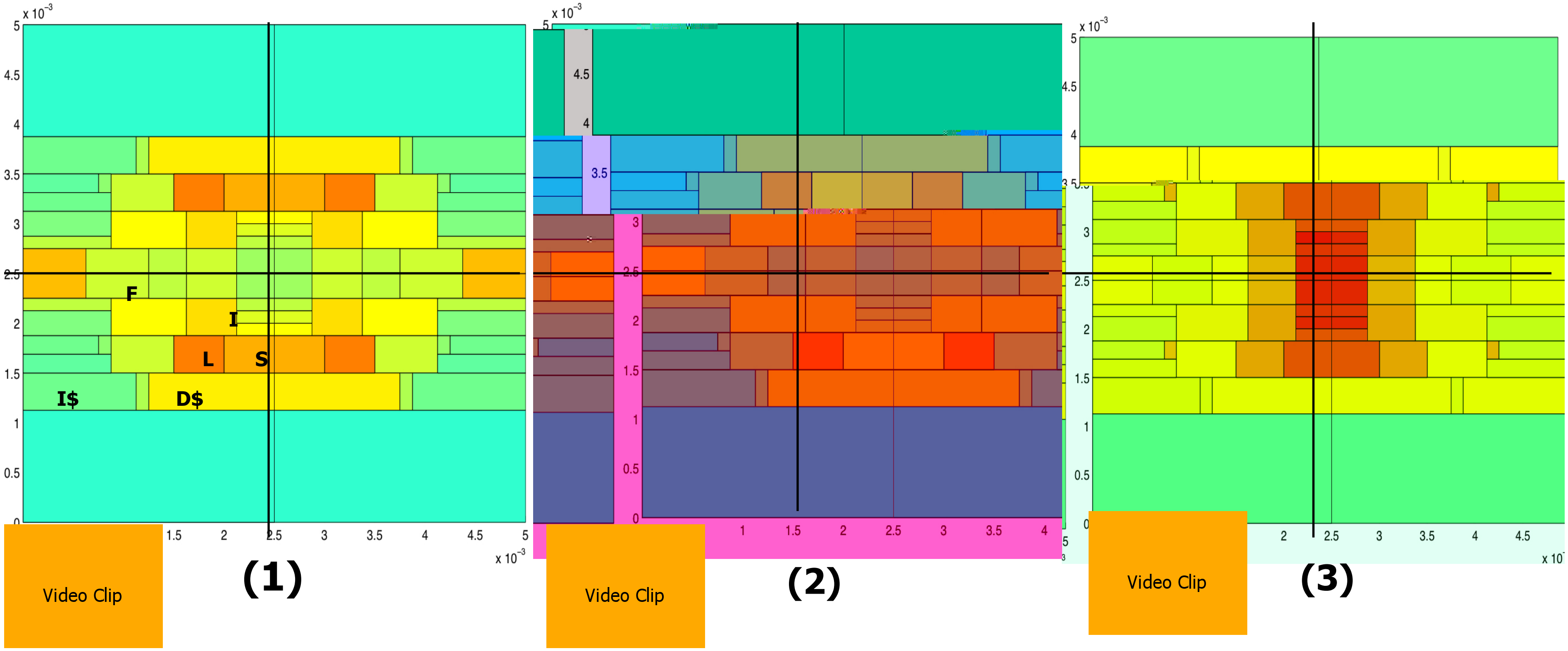
Video Clip

(3)

Temp: 0   **65 C**

Temperature Evolution (4-core)

SMV_RCM **DGEMM** **SMV_RND**

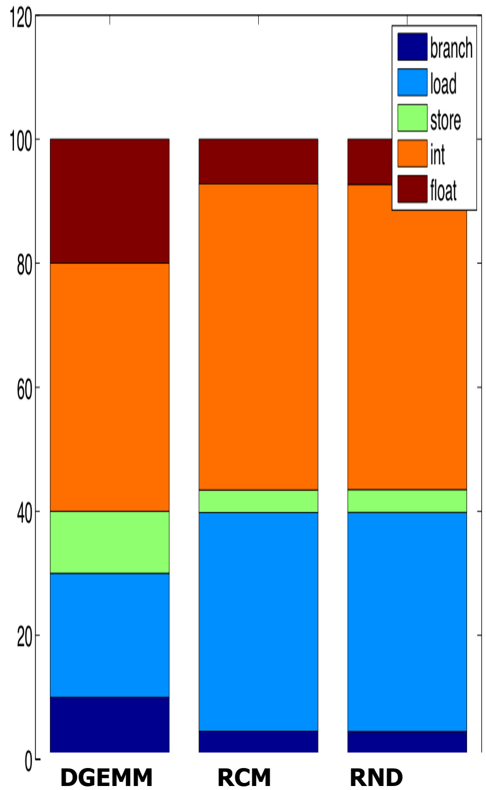


(1) **(2)** **(3)**

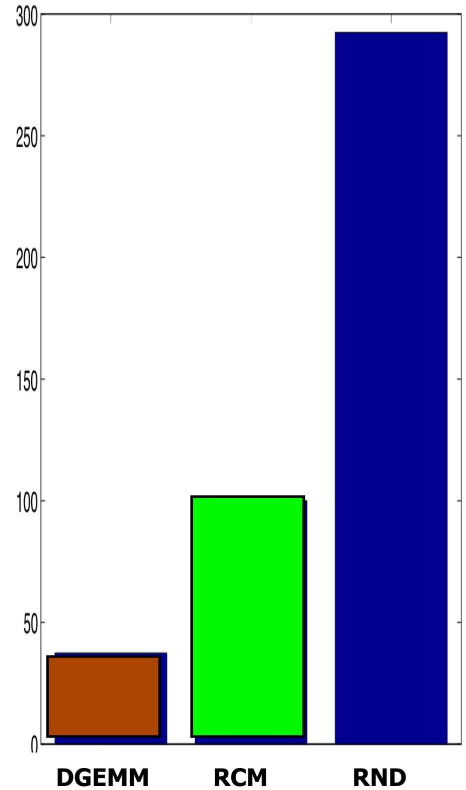
Temp: 0 **65 C**

DGEMM, SMV Profiles

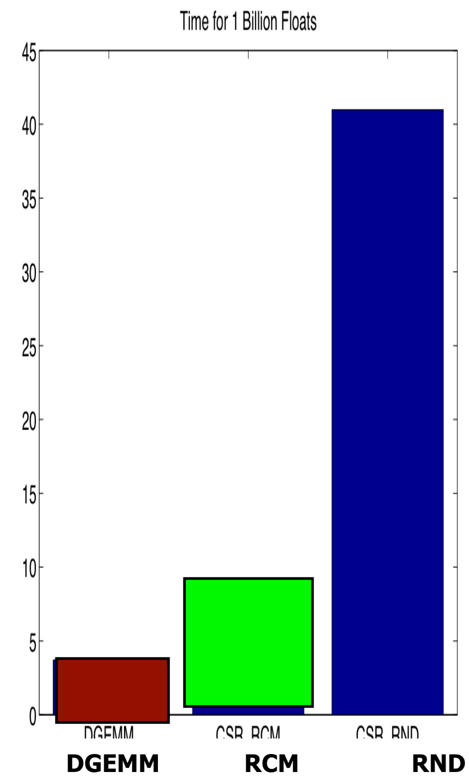
Instruction Mix



Energy/ 1 B Floats

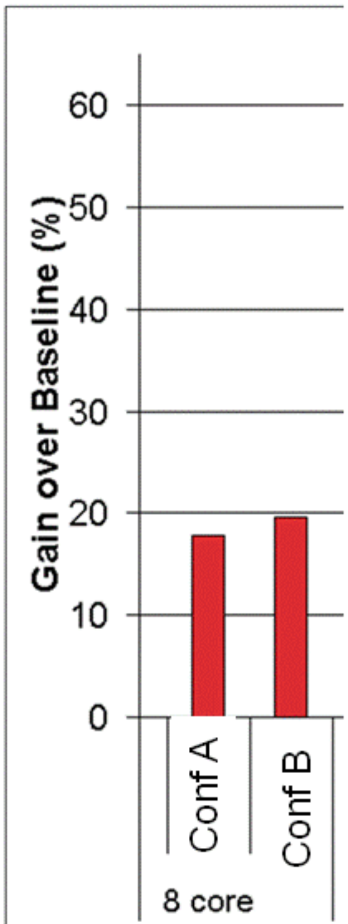


Time / 1 B Floats

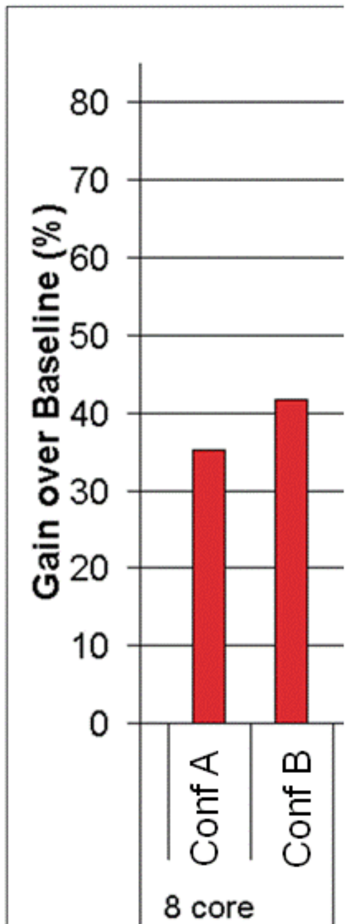


Cache vs SPM: MatVec

Dense MatVec
Relative Improvements (8 cores)

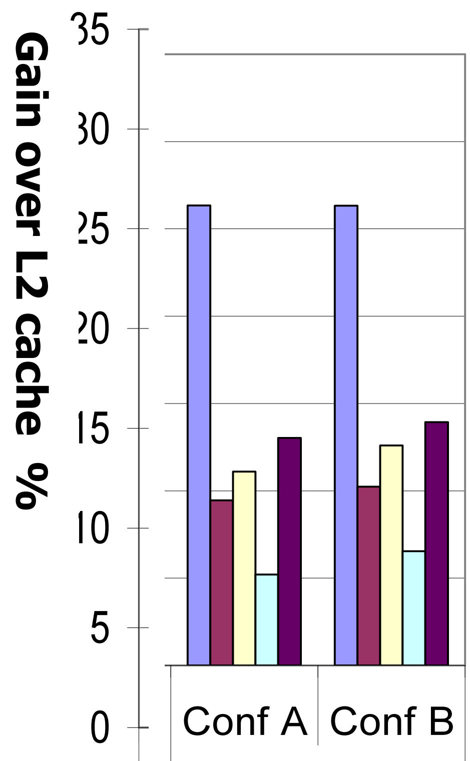


Performance

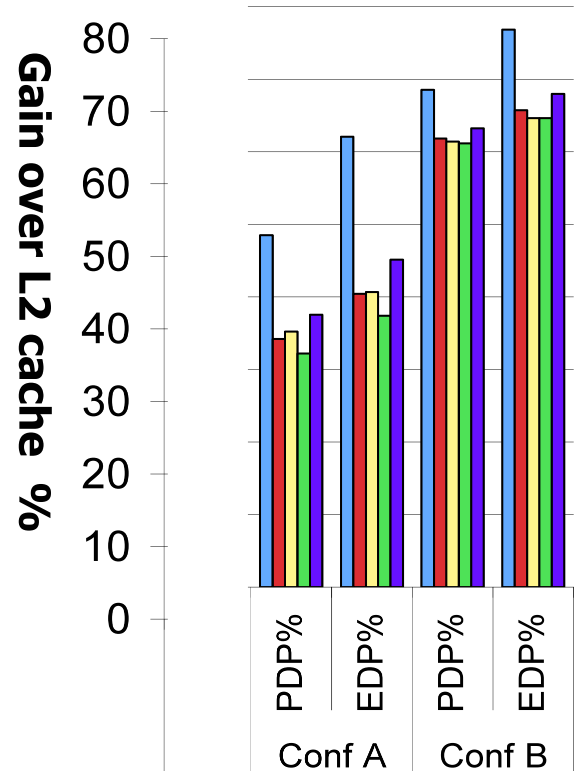


Energy (EDP)

Sparse MatVec
Relative Improvements (8 cores)



Performance



Energy (EDP)

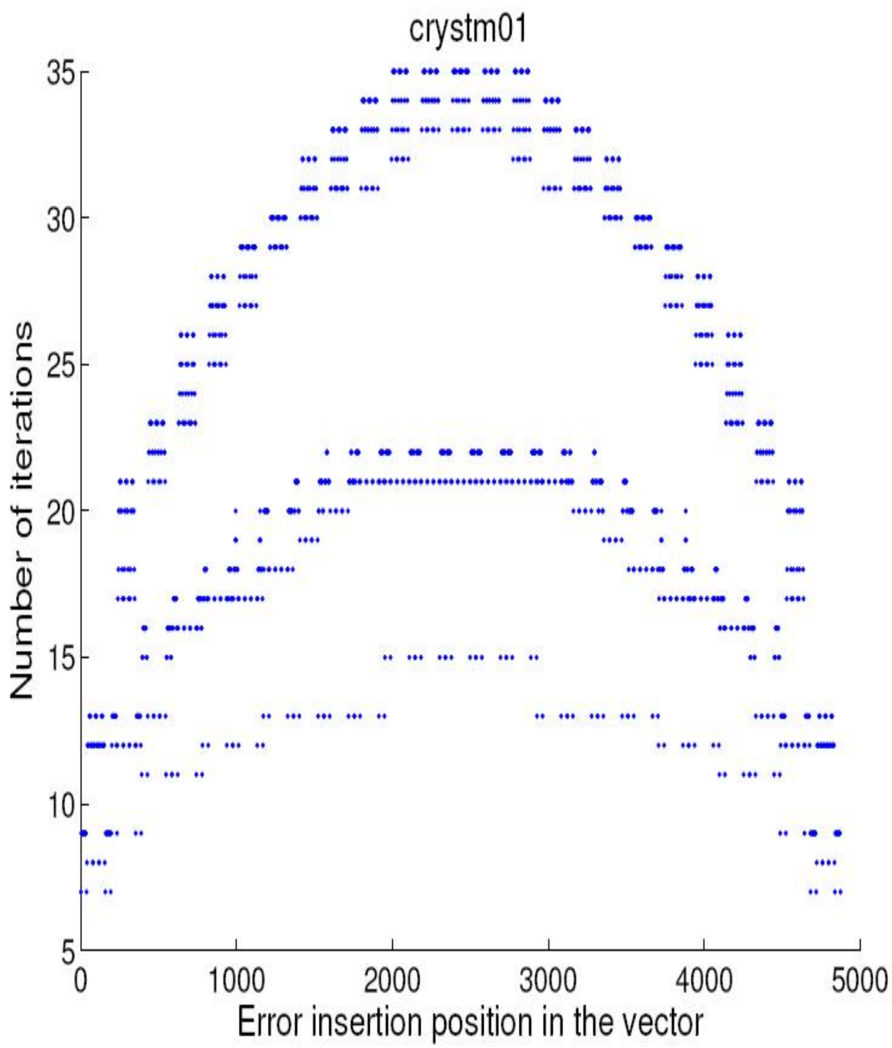
3. Hard Facts about Soft Errors

0	0	0	0	1	0	0	0
1	0	1	0	1	0	1	1
0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	1
0	1	0	1	0	1	0	1
0	1	1	1	0	1	1	1
0	1	0	1	0	1	0	1
0	0	0	0	1	1	0	1

1. Soft (transient) errors from timing, thermal, alpha particle strikes
2. Single cell upset rate ~ 0.001 FIT (1 failure in 1 billion hours)
3. 16KB L1 - 100,000 cores : 1 failure every 76 hours.

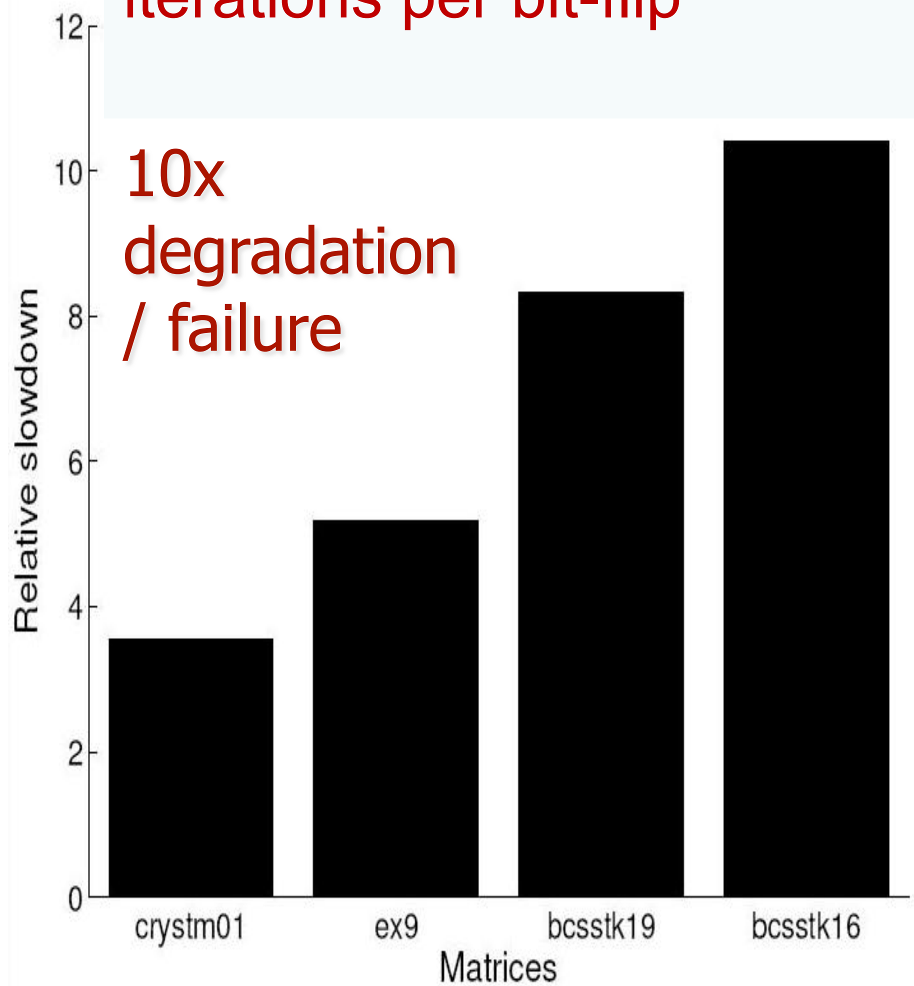
Malkowski et al. IPDPS 2010

Effect of Single Bit-Flip on PCG

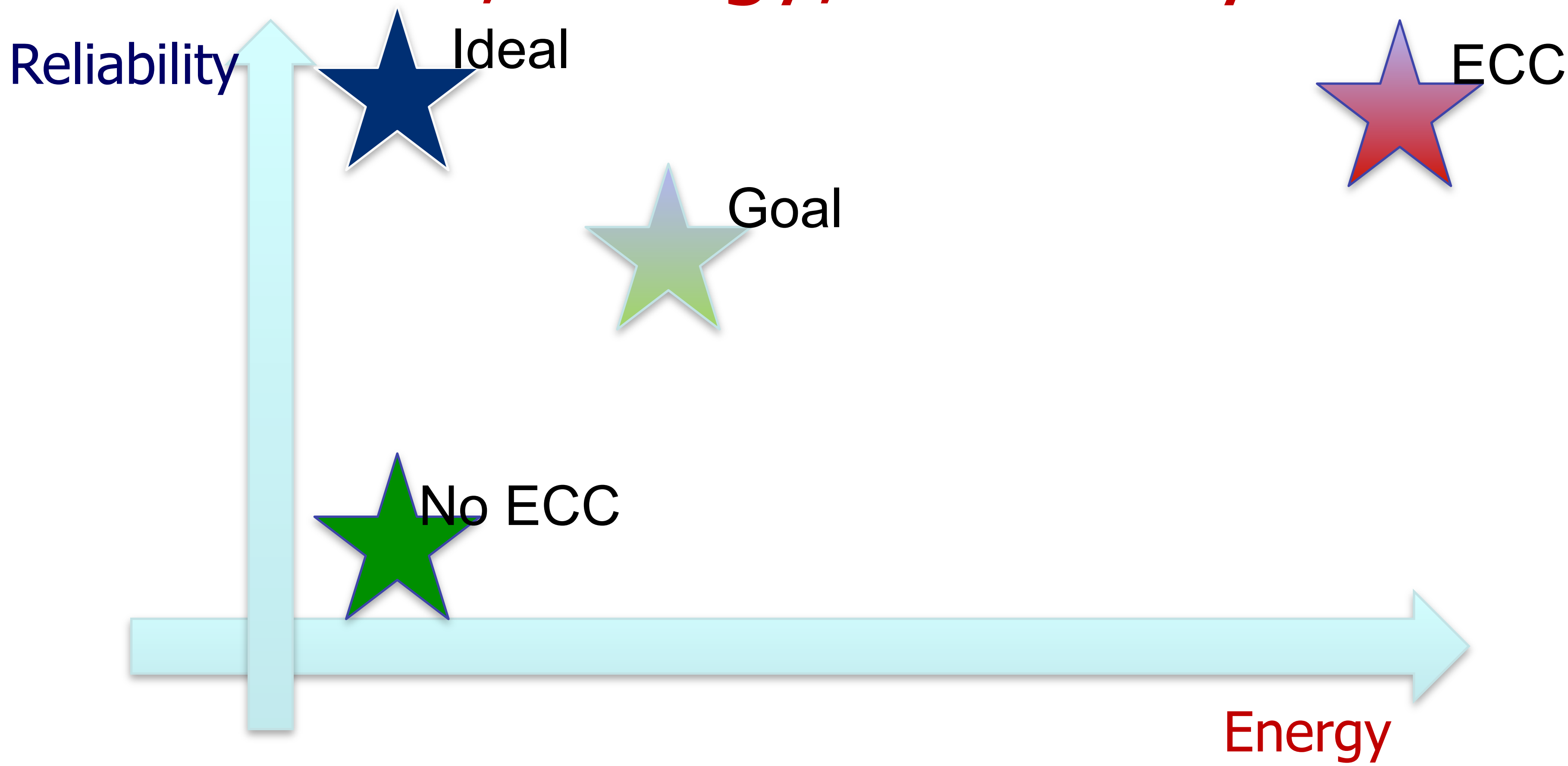


Ratio of max-min PCG iterations per bit-flip

10x degradation / failure



Trade-offs:
Performance, Energy, Reliability



Algorithm, s/w, h/w methods to find sweet spot in performance, energy & reliability tradeoffs

Managing Tradeoffs

- S-W/H-W methods
  - Programmer provides hints
    - on data structures that require strong ECC, low ECC, no ECC protection
- Algorithmic approaches
  - Adapt iterations to detect/correct convergence
  - Voting methods
  - Encoding methods

Summary

- Payoffs in exploiting high dimensional parameter space
  - to manage performance, energy, reliability trade-offs
- Run time systems for S/W -H/W cross-layer optimizations
- Monitor-Model-Adapt frameworks for on-line (and off line) auto-tuning & optimizations
- Cores are cheap and be used to "auto adapt"
- Multicores as a bridge and driver for change
  - big vs. small science,
  - sparse/irregular vs. dense/regular,
  - commercial vs. research