(1)Mini-Ckpts: Surviving OS Failures in Persistent Memory
(2) Ptune: Power Tuning HPC Jobs

Frank Mueller

North Carolina State University

in collaboration with

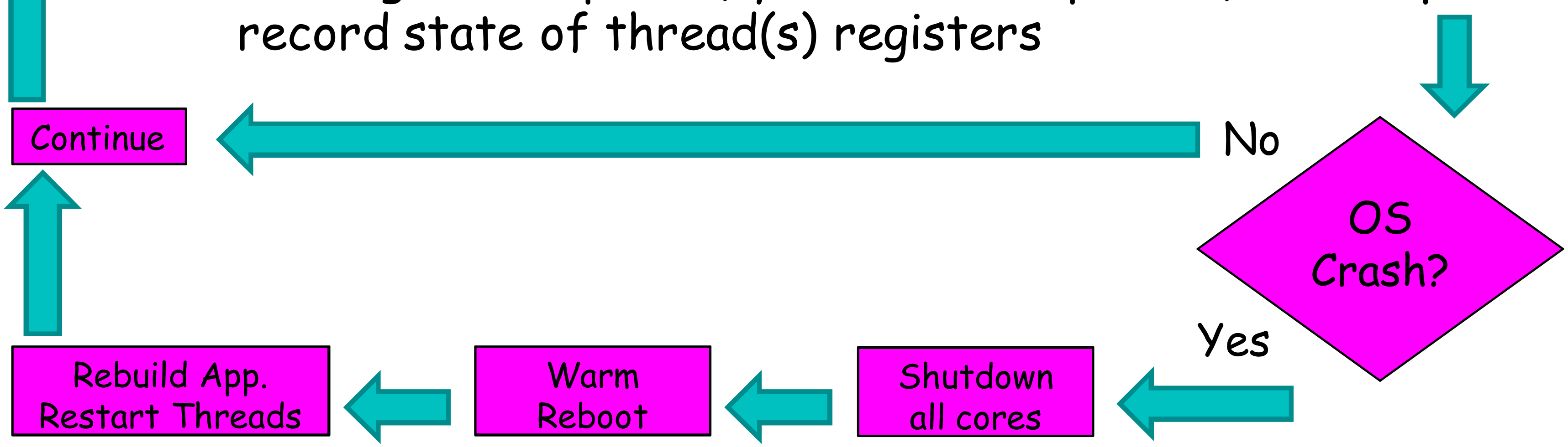
ORNL, SNL, LLNL

(1) Mini-Ckpts: Protect the Operating System

- **Why protect OS?** → Any failure causes "panic", loss of all unsaved computation. OS remains the last unprotected piece
- Objective: Let app survive if OS fails, recover OS quickly
- Design of Mini-Ckpts:
  - Identify minimal process state @ failure
  - Identify common instrumentation points in OS to save state
  - Warm reboot OS on failure, preserve app and continue exec.
- Implementation:
  - Process protection from kernel failures at syscalls
  - App lives in persistent memory
- Evaluation:
  - cost of mini-ckpts and warm-rebooting a failed OS
  - application survival for injected kernel faults (OpenMP+MPI)

Mini-ckpts Overview

- Requires specialized kernel → new NMI for panic shutdown
- Requires persistent memory → Linux PRAMFS
- Protection
  - Checkpoint (serialize) structures describing a process
  - Migrate memory to persistent region (survives warm reboot)
  - continue execution…
    - During interruption (syscall, interrupt IRQ, interrupt NMI) record state of thread(s) registers



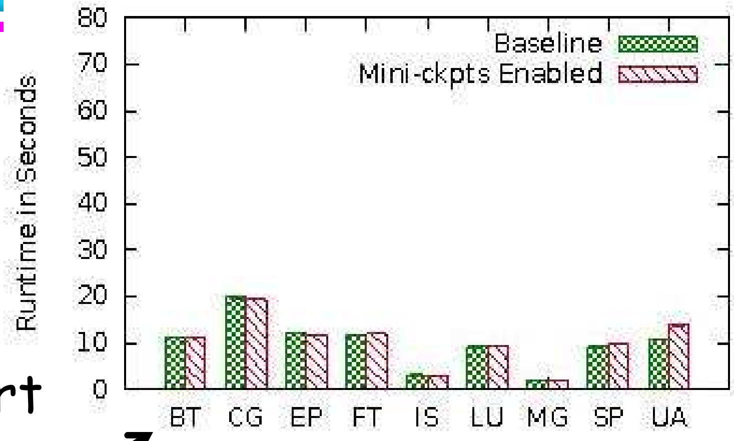
Warm Reboot

- Time from kernel panic until
  - (a) kernel is loaded, and
  - (b) software stack initialized from PRAMFS
    - Single largest kernel boot cost: network initialization
- Warm Reboot Total → time at which app may be restored/resumes
- Virtual machines (VMs) do not require initializing physical h/w
  - i.e., network cards

(measured in seconds)	BIOS Boot Time	**Kernel Boot Total**	Network Driver & NFS-Root Mounting	Kernel Misc	Software Stack Total	Cold Total w/ BIOS	**Warm Reboot Total**
AMD Bare Metal	37.4	**5.3**	1.5	4.8	0.7	50.3	**6.0**
Intel Bare Metal	50.8	**6.7**	3.0	3.7	0.7	73.0	**7.4**
AMD VM	—	**0.8**	< 0.2	< 0.6	3.0	—	**3.8**
Intel VM	—	**0.7**	< 0.2	< 0.5	1.3	—	**1.9**

Experiments (Excerpt)

- We make apps resilient if OSfails
- Rejuvenates kernel,
  apps survives in persistent memory
- Ckpt/restart is expensive for HPC apps
  - OS crash → **fwd progress** w/o restart



- Warm reboots in ~6 seconds
  - 5%-8% overheads, threaded+MPI apps, scalable in # threads

(measured in seconds)	BIOS Boot Time	**Kernel Boot Total**	Network Driver & NFS-Root Mounting	Kernel Misc	Software Stack Total	Cold Total w/ BIOS	**Warm Reboot Total**
AMD Bare Metal	37.4	**5.3**	1.5	4.8	0.7	50.3	**6.0**
Intel Bare Metal	50.8	**6.7**	3.0	3.7	0.7	73.0	**7.4**
AMD VM	—	**0.8**	< 0.2	< 0.6	3.0	—	**3.8**
Intel VM	—	**0.7**	< 0.2	< 0.5	1.3	—	**1.9**

Mini-Ckpts Summary

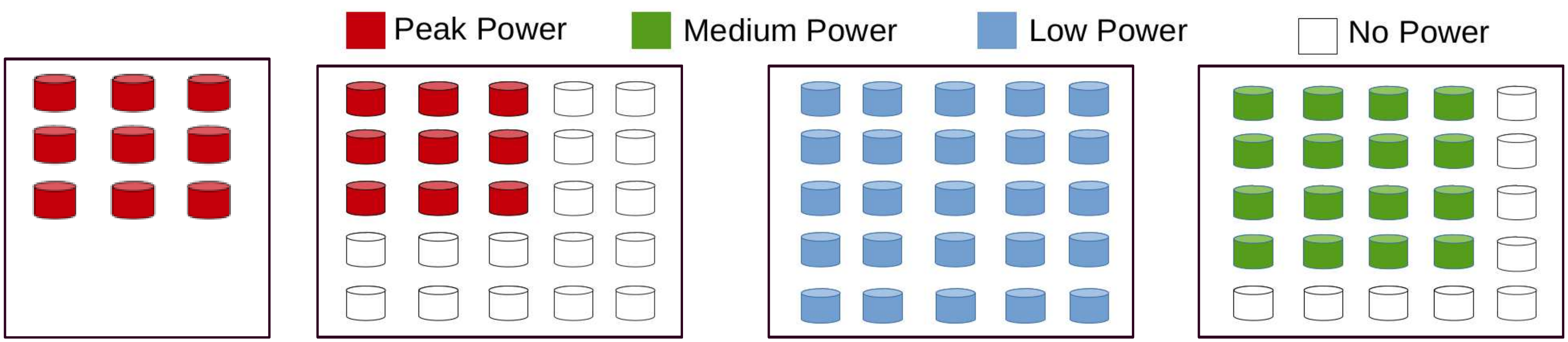
- Today's OS's not designed with fault tolerance in mind
  - Mini-ckpts provides resilience to appliations if kernel fails
  - Rejuvenates kernel, apps survives in persistent memory (PRAMFS)
- Ckpt/restart is expensive for HPC apps
  - mitigating an OS crash allows **forward progress** w/o restart
- Mini-ckpts identifies key OS changes & structures req'd for resilience
- Warm reboots complete in ~6 seconds, overheads between 5%-8%
  - Both threaded and MPI applications recoverable
  - Scalable in # threads

1st ever transp. OS fault tolerance w/o loss of state

Apps could outlive OS → even if OS instable

(2) Ptune: Power Tuning HPC Jobs

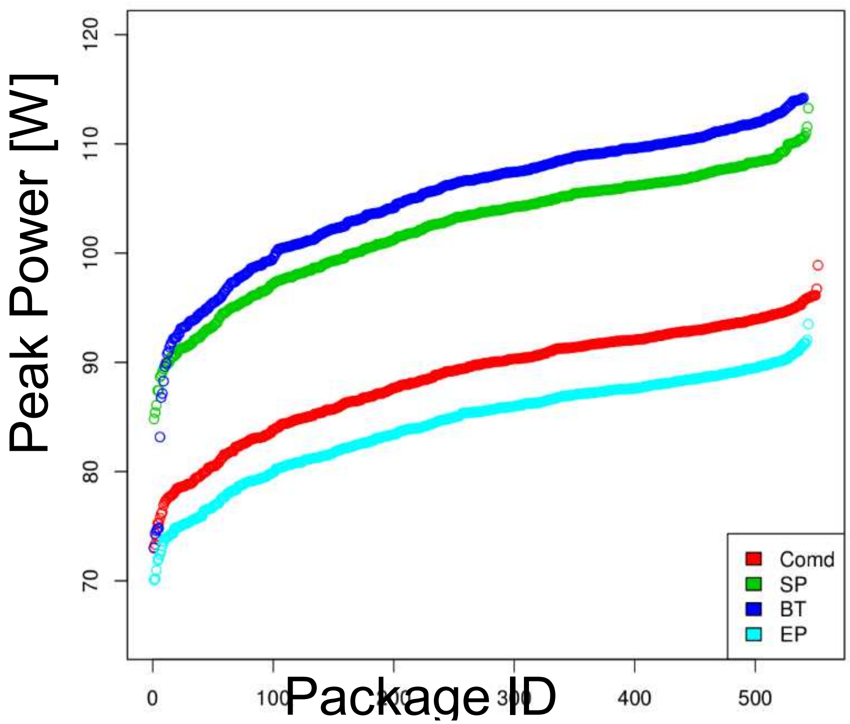
- **Target : Exascale by 2020 i.e. 2^{18} FLOPS**
- Today: Sunway TaihuLight
  - 93 PFlops ~ **0.1 Exaflops w/ 15.37 MW** → **1 Exaflops w/ 150MW?**
  - US DOE power budget of **20MW per exaflop system**
- need order of magnitude improvement in performance + power together!
  - **Goal : Maximize(Performance per Watt)**
- Today: only 60% of the procured power used after Linpack burn-in
- **Solution: Hardware overprovisioning**
  - Buy more nodes than can be powered

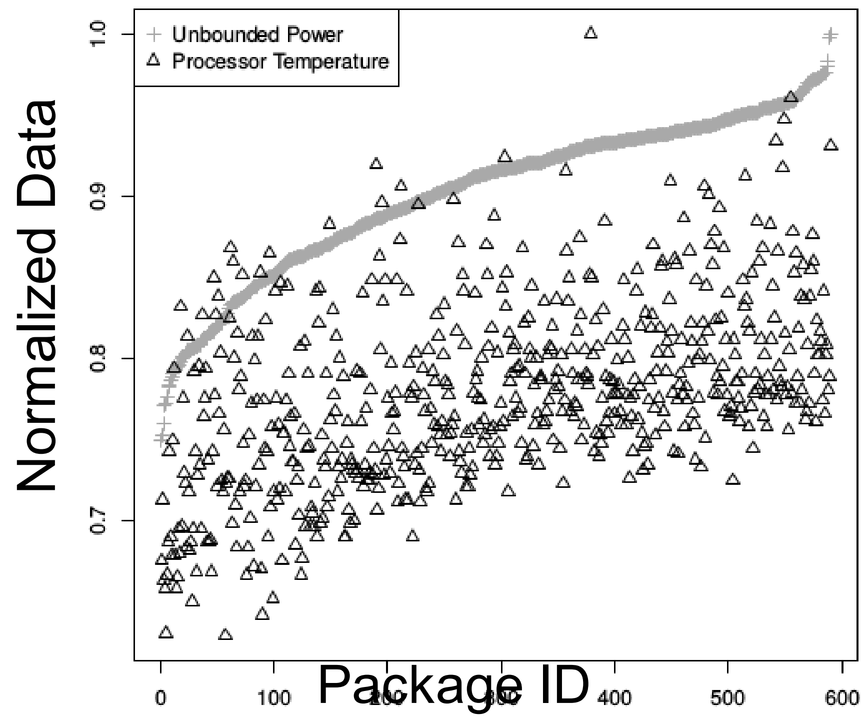


WCP

Processors Vary in Power Draw

- under fixed performance
- Packages from the same Stock Keeping Unit (SKU)
- Peak Performance: Uniform; Peak Power: 30% Variation
- Potential Causes: Process variation, Thermal variation, etc.

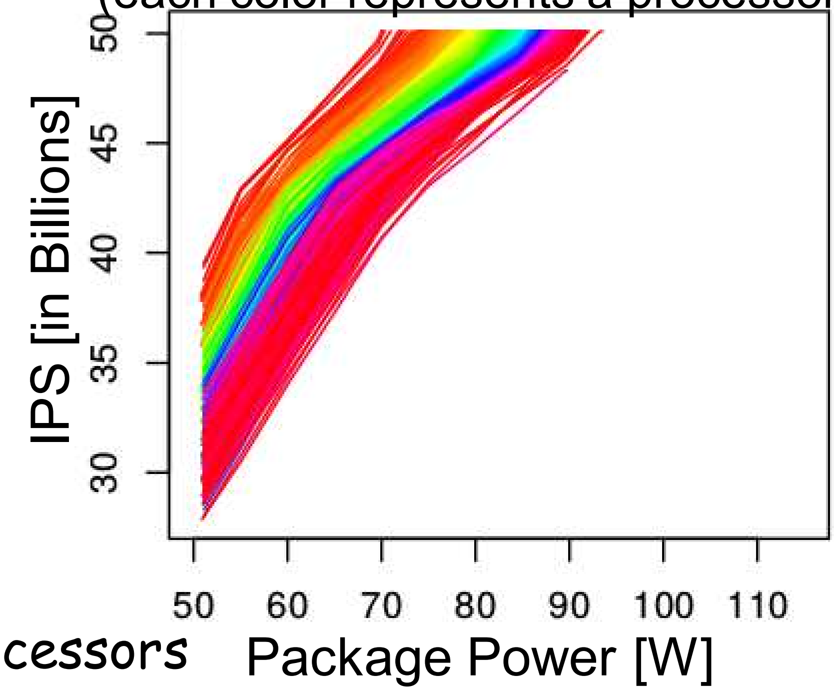




Performance Variation with Power Caps

- Intel: Running Av.g Power Limit (RAPL):
  - **PKG (processor)**
  - DRAM
- measures avg. power short period
- can set power cap
  → will never exceed this level

(each color represents a processor)



- Data for 600 Intel Ivy Bridge processors
- Performance : Instructions Per Second (IPS)
- **30% Performance Variation**

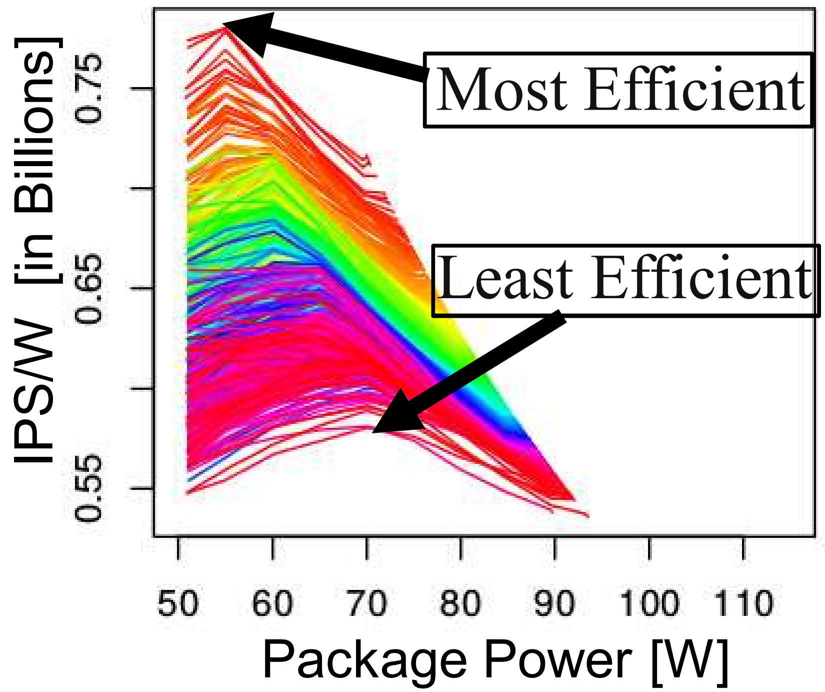
> **Variability in peak power → Performance variation**
> **Power-Performance curves differ from application to application**

Power Efficiency

- *power efficiency* := instructions retired per second (IPS) per Watt
- Variation in peak power efficiency
- Less efficient processors are most efficient at higher power bounds

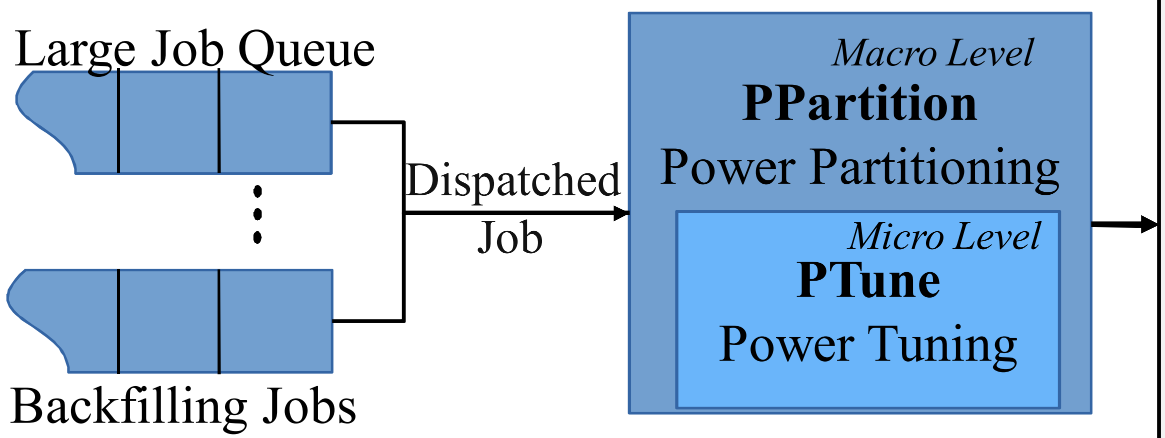
Variability in peak power → Variation in peak power efficiency

(each color represents a processor)



Power efficiency curves differ from application to application

Power-aware job scheduling and tuning



For all jobs, determine:

1. job's power budget, P_{Ji}
2. n_{opt}
3. selection of n_{opt} processors
4. power distribution across them

- n_{opt} : optimal # of processors for a job under its power budget

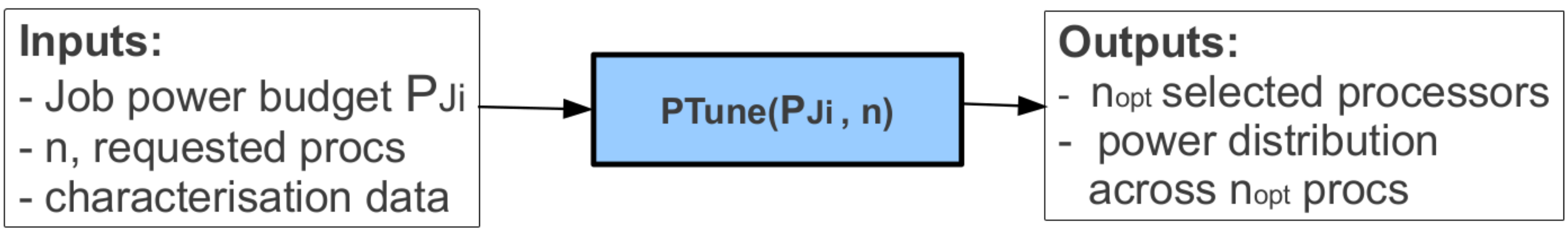
Assumptions:

- Hardware overprovisioned system w/ strict power constraint
- Now: limited to CPU power (extensible)
- moldable jobs: can vary # processors for app

PTune at job level

Goal: Maximize(JobIPS) under fixed job power budget

- Choose **n_{opt} processors** from available ones
- Determine **non-uniform power distribution** for job across n_{opt} procs
- Greedy: return unused expensive (inefficient) processors back to pool of unused processors for other jobs

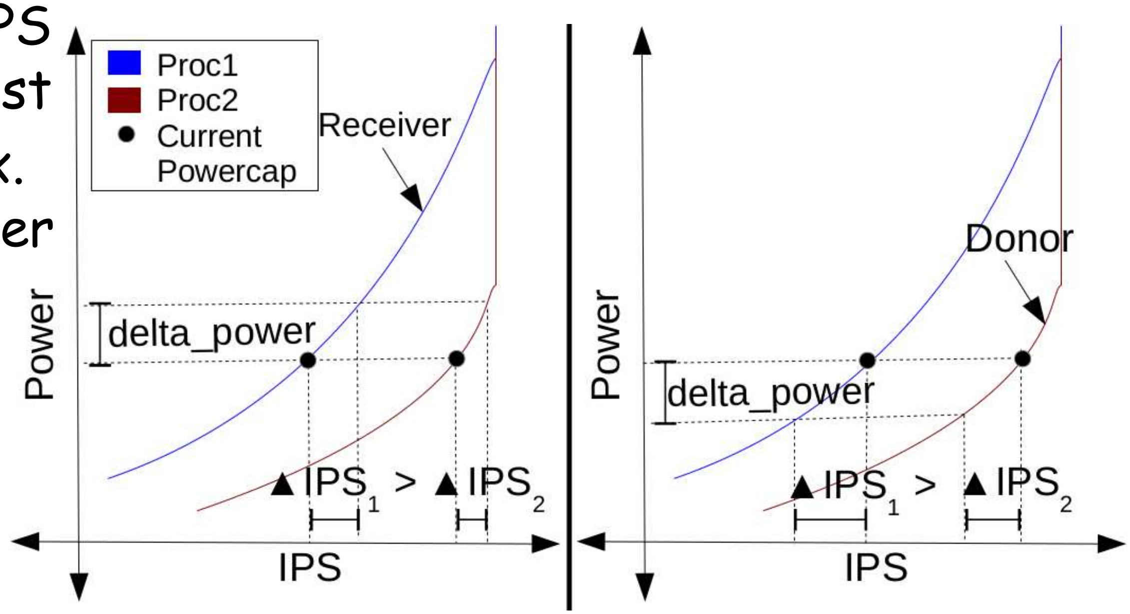


- **Step 1:** Sort available procs by power efficiency (a priori, once)
- **Step 2:** Add n^{th} proc by stealing power from former (n-1) procs

PPart at scheduler level

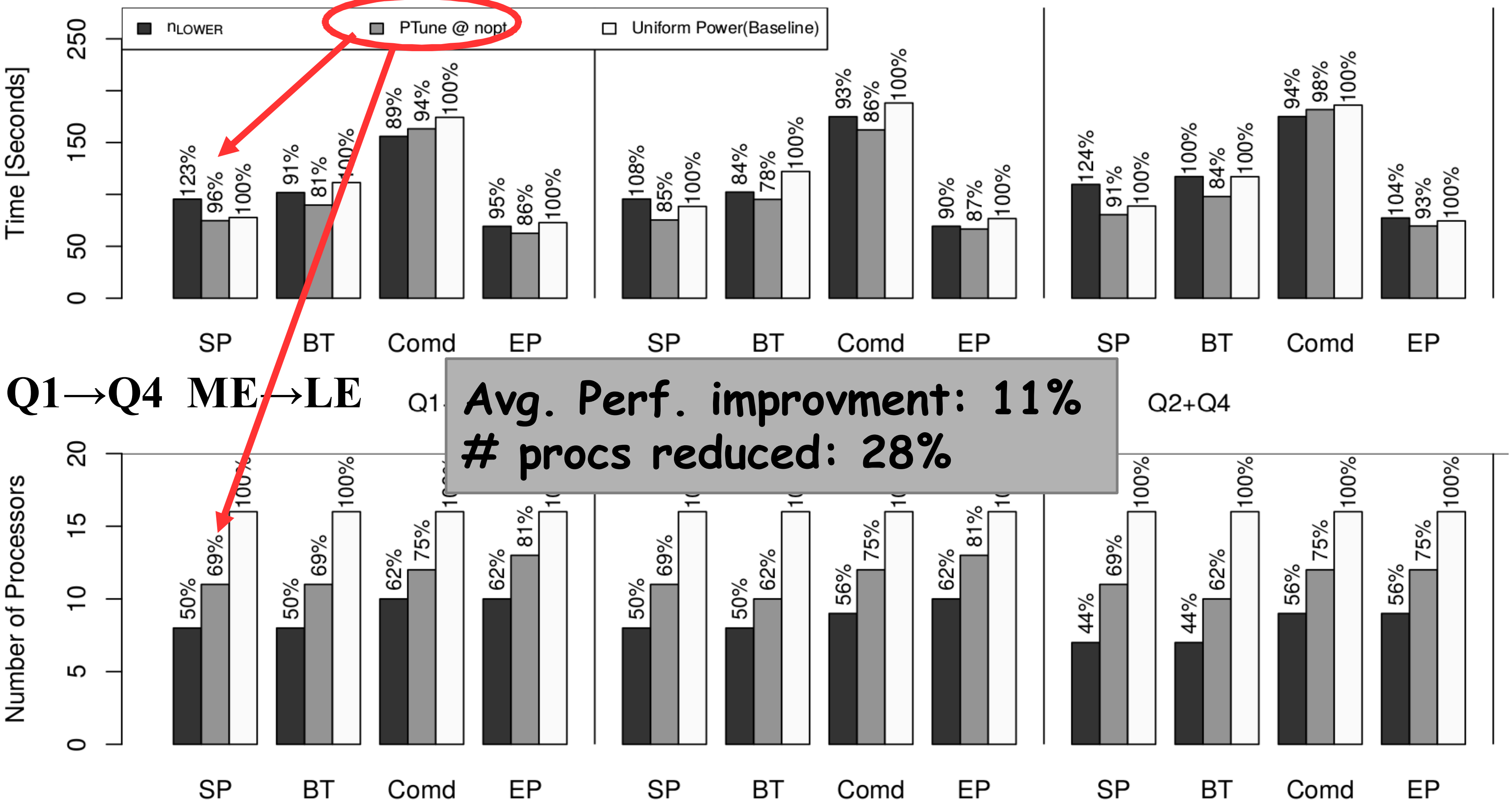
Goal: Maximize(JobIPS) under fixed job power budget

- power re-partition when new job dispatched by scheduler
  - works across jobs
- **If P_{Ji} is available → *Tune Ji for P_{ji}***
  Else *Steal Power from already scheduled jobs*
  - Donor : proc w/ min. IPS loss for delta power lost
  - Receiver : proc w/ max. IPS gain for delta power

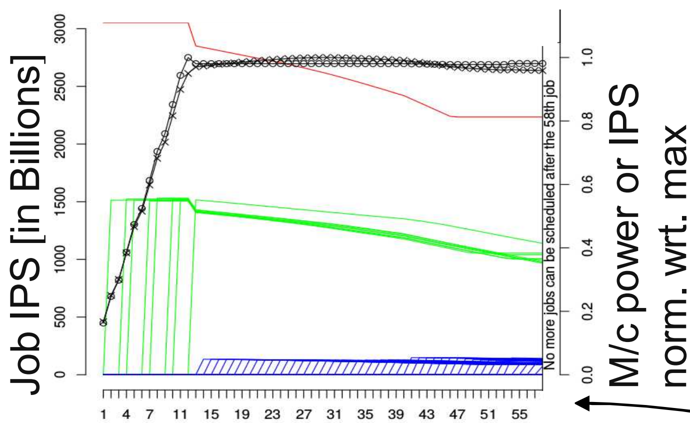


PTune 16 packages (Power Budget = 8KW)

Performance (Improvement) depends on - 1. processors (**variations observed under cap**) and 2. application **(more/less IPS/W)**



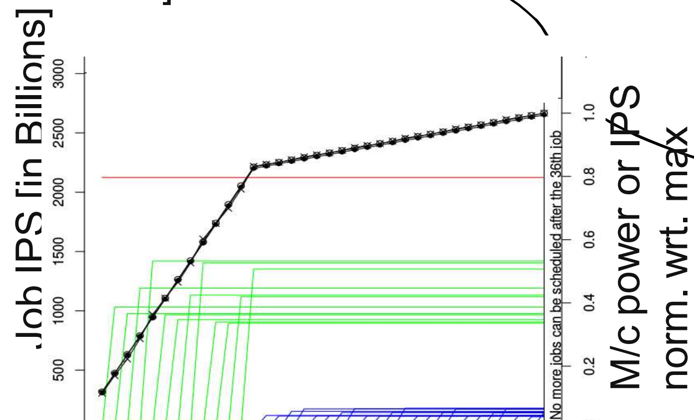
PPart results for Pm/c = 28KW

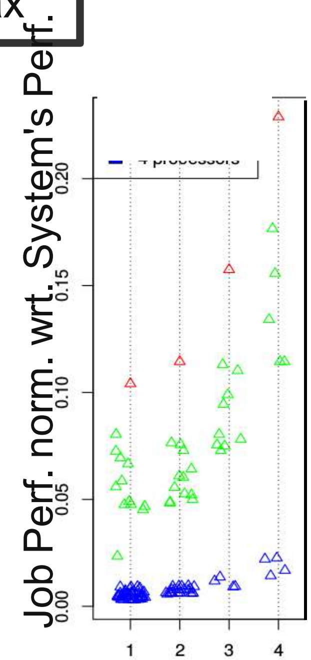


64 processor
32 processor
4 processor

1-PTune+PPartition (Ref.)
2-Uniform Capping at $P_{m/c}/N_{max}$
3-Uniform Capping at 75W
4-Uniform Capping at TDP

M/C power norm. wrt. Pm/c
M/C throughput norm. wrt. max

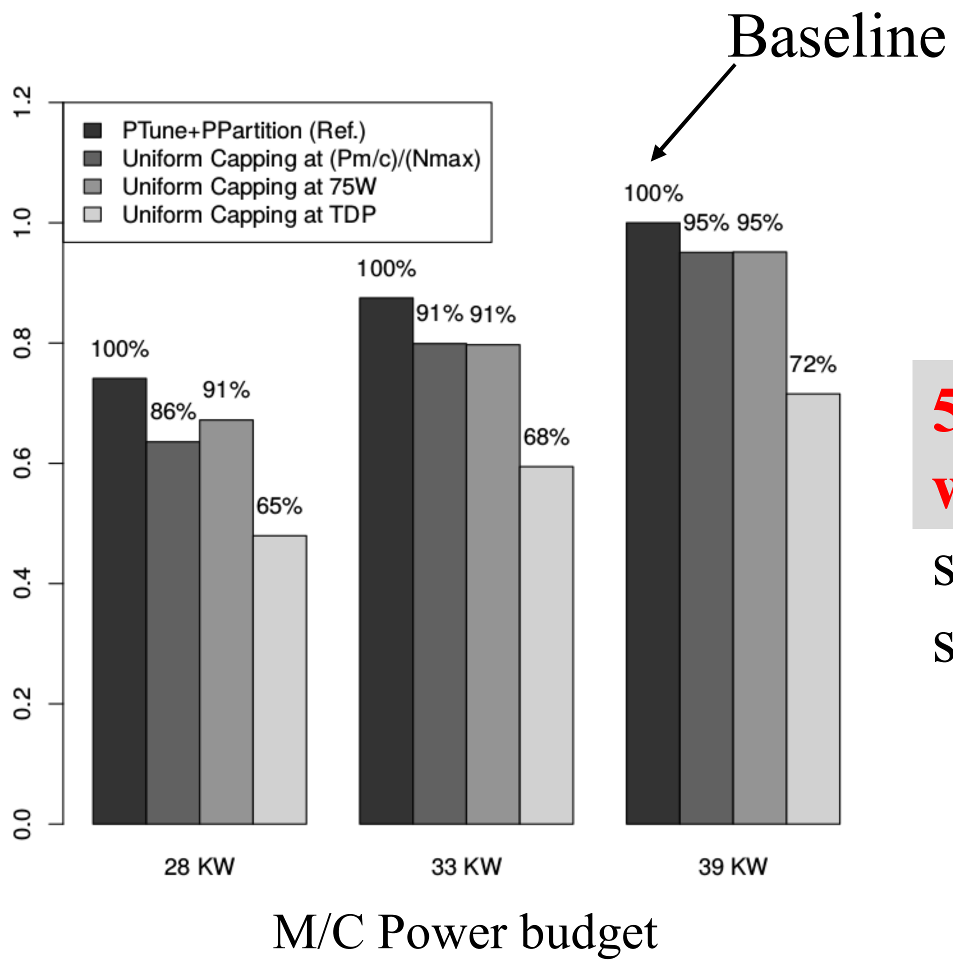




More jobs scheduled with PTune+PPart **under the same Pm/c**

PPart Results

- Simulated System: Nmax = 550 procs w/ 28KW, 33KW, 39KW



5-35% improvement with PPart+PTune solution over the naive scheduling schemes

Conclusions

- **ICS'16 miniCkpts: apps survice OS crashes in persist. memory**
  - Warm reboots in ~6 seconds, overheads between 5%-8%
- PACT'16: Power efficient HPC operations via power caping
  - **Ptune:** 29% improvement in job performance vs. uniform power
  - **PPart+Ptune:** improve job throughput by 5-35% vs. naïve scheduling w/ power budget
- **IPDPS'16 TintMalloc:** controller+LLC-aware alloc.for threaded codes
  - *Avoid remote memory* node access
  - *Reduce bank+LLC conflicts*
  - Parallel tasks
    - ***Up to 75% more balanced*** */ less idle time @ barriers*
    - ***Up to 30% higher performance*** */ reduced runtime*
  - Better than "Standard Buddy + numa library"
  - Only 1 additional line of code: 1 mmap() call @

Acknowledgement

Supp. in part by DOE/NFS grants, Humboldt fellowship

DOE DE-FG02-05ER25664, DE-FG02-08ER25837, DE-AC05-00OR22725, NFS 0237570, 0410203, 0429653, 1058779, 0958311, 0937908

DOE DE-AC04-94AL85000 (SNL) , DOE DE-AC05-00OR22725 (ORNL) , LBL-6871849 (LBL)

sponsored by the U.S. Department of Energy's Office of Advanced Scientific Computing Research

- NCSU: David Fiala, Neha Gholkar, Frank Mueller
- ORNL: Christian Engelmann
- SNL: Kurt Ferreira
- LLNL: Barry Rountree

Lawrence Livermore National Laboratory

NC STATE UNIVERSITY Department of Computer Science

Sandia National Laboratories

OAK RIDGE National Laboratory

NORTH CAROLINA STATE UNIVERSITY FOUNDED MARCH 7, 1887