Abstract

• As scientific applications target extreme scales, energy-related challenges are becoming dominating concerns. As a result, it is critical to explore emerging architectures (e.g., with multiple cores and deep memory hierarchies) and applications (e.g., coupled simulation workflows) from an energy perspective and investigate associated overheads and tradeoffs. For example, energy/power-efficiency have to be addressed in combination with quality of solution, performance and reliability, and other objectives, and achieving the desired levels of reduction in power consumptions requires a comprehensive cross-layer and application-aware strategy. In this talk I will explore these issues and will describe recent related research efforts at the Rutgers Discovery Informatics Institute (RDI2).
Outline

• Power/Energy Challenges at Extreme Scale
  – Research landscape

• The GreenHPC Project @ Rutgers
  – Application-aware power management
  – Programming support for GreenHPC
  – Power/Energy-aware data management

• Concluding Remarks and Open Challenges

Rutgers Discovery Informatics Institute (RDI²)
Driving Innovation through Advanced Computing

- Established in March 2012 as New Jersey’s Center for Advanced Computation with an overarching goal to create a world-class institute focused on computational and data sciences
- Fundamentally integrate research, education, ACI and industry partnerships to address core CDS&E / BigData challenges
- Broaden industry and academic access to state-of-the-art computing technology and expertise
  - NSF Cloud and Autonomic Computing IUCRC
- Integrate multidisciplinary research with ACI and industry partnerships

http://rdi2.rutgers.edu
Modern Science & Society Transformed by Compute & Data

- New Paradigms & Practices
  - End-to-end: Seamless access, aggregation, interactions
  - Data-driven, Data/Compute-intensive; Age of Digital Observation
  - Integrative, multi-scale, online

- Multi-disciplinary collaborations
  - Individuals, groups, teams, communities, networks
  - New global science culture

- Unprecedented opportunities, challenges

Advanced Computing Infrastructure

- Large scale, distributed, heterogeneous, multicore/manycore, accelerators, deep storage hierarchies, experimental systems

Titan - Cray XK7
- 20 PF / 300 K cores
- 16-core CPU + GPU
- Gemini 3D torus
- 600 TB memory

Sequoia – IBM BG/Q
- 20 PF / 1.5 M cores
- 18-core processor
- 5D torus
- 1.5PB memory

Worldwide LHC Computing Grid
- >140 sites;
- ~250k cores;
- ~100 PB disk

XSEDE
- Worlds Largest Grid
- 11 Resource Providers

Modern Datacenters
- 1M servers
- 50-100 MW

Special Purpose HW (Anton)
- > 100 time acceleration of MD simulations
Energy Efficiency Critical at Extreme Scales

- Power has become a critical concern for HPC/supercomputing
  - Impacts operational costs, reliability, correctness
  - End-to-end integrated power/energy management essential
- Increasing scale towards exascale
  - Using existing technology would require gigawatts
    - Multiple nuclear reactors
    - > $2.5B annual power cost
  - Target < 20MW !!

- Power/energy not only from computation
  - Data transfer
    - Memory, I/O, HPC network (e.g., 3D torus)
  - Data storage
    - Disk, NVRAM, etc.

Managing Energy/Power – A Wide Range of Techniques

- Aggressive component-level power management
  - CPU (e.g., DVFS), memory, disk, NIC, etc.
- Run-time power management
  - System level, exploiting slack in MPI programs, etc.
- Application/workload power management
  - Workload profiling/characterization, consolidation, etc.
  - Application, algorithm adaptations
- Autonomic policy adaptation
  - Characterize operational state and adjust management goals
- Cooling and thermal management
  - React to thermal hotspots, proactive workload placement
  - …, etc.

Can we use these together in a cross-layer, coordinated, and consistent manner?
Green HPC: Landscape (I/III)

- Dynamic use of low power modes (sleep, suspend, hibernate)
  - Advanced Configuration and Power Interface (ACPI)
    - Pallipadi et al. [OIS'07]
  - Switching on/off servers for energy conservation
    - Heath et al. [PPoPP'05]

- Dynamic scaling (usually impacts latency)
  - Processor Dynamic Voltage and Frequency Scaling (DVFS)
    - During communication (MPI slack)
      - Kappiah et al. [SC'05], Lowenthal et al. [SC'06, SC'07], Freeh et al. [IPDPS'05, PPOPP'06]
  - Using application profiles
    - Cameron et al. [SC'05, Computer'05]
  - Using counters
    - Hsu et al. [SC'05], Rountree [ICS'09]

Green HPC: Landscape (II/III)

- Dynamic scaling (contd.)
  - Scaling other components/subsystems
    - Dynamic memory frequency
      - Delaluz et al. [IEEE TC'01, DAC'02], Fradj et al. [DSD'06], Bianchini et al. [ASPLOS'11]
    - Storage subsystem -- Multi-speed disks and RAIDS
      - Rotem et al. [IPDPS'09], Pinheiro et al. [ICS'04]
    - NVRAM -- PCM/STTM, memristors, flash-based
      - Caulfield et al. [ASPLOS'09, MICRO'10], Bianchini et al. [SC'11]
    - Flash-based SSD
      - Urgaonkar et al. [OSDI'08]
Green HPC: Landscape (III/III)

- Other approaches
  - Architecture/Instruction-set based
    - Martonosi et al. [ISCA’00, ISCA’01, HPCA’12]
  - Cluster-based power management controller
    - Ranganathan et al. [SIRARCH’06]
  - Job allocation and scheduling for power conservation
    - Chase et al. [SOSP’01], Pinheiro et al. [DCS-TR-440], Zong et al. [ICPP’07]
  - Network power management
    - Lefevre et al. [IPDPS’09, ISPA’11]
  - Thermal-aware
    - Skadron et al. [TACO’04]
  - Virtualized environments, programming languages, etc.
  - ...

Energy/Power Management at Multiple Layers

- Application Design & Execution
  - Algorithms design, adaptive execution
- Power-Aware Compilers and Runtime Support
  - Power-aware code generation and scheduling
  - DVFS, programming models, abstractions
- Operating System level
  - Power control (e.g., ACPI)
- Architecture
  - Processor, memory, storage, and interconnect technologies
  - Increased use of accelerators
  - Power-aware architectures
- Infrastructure – sensing environment
  - Infrastructure: thermal sensors, instrumentation, monitoring, cooling

Application-aware & Cross-layer

- Cross-layer optimizations
  - Independent optimizations may not be effective / sufficient
  - Conflicting goals/policies/actions at different layers
  - Need to look at entire system
  - Multidimensional tradeoffs – energy vs. reliability vs. performance vs. application quality ….

- Application/workload-aware optimizations
  - E.g., adapt resources to application / application to the resources
  - Address unique challenges, leverage unique characteristics of typical HPDC applications
  - Models, abstractions, mechanisms, policies, APIs, for aware cross-layer management
GreenHPC – Overarching Research Goals

• Multiple orders of magnitude power/energy reduction are required – a cross-layer and application-aware strategy is necessary
  – Innovations necessary at multiple levels (algorithms, programming, runtime, OS, …)

• Energy/power-efficiency in combination with other objectives – Tradeoffs are essential
  – Quality of solution, Performance, Reliability, etc.

• Transporting, processing and analyzing increasing data volumes and rates is becoming the dominant challenge

GreenHPC @ Rutgers – Selected Research Efforts

1. Application-aware power management
   – Using application behavior for aggressive proactive management

2. Programming Support for GreenHPC
   – Application-driven management using hints

3. Power-aware data management
   – Managing power for end-to-end application workflows
1. Application-aware power management

Application-aware Power Management

• Objective: Use application behaviors to enable aggressive and proactive power management
  – Reactive power management is not always optimal and can result in large overheads
  – Proactive adaptation of resources based on application behavior and requirements (e.g., subsystems demand over time)
  – Maintain performance to the extent possible

• Overall approach [HiPC10, HiPC11]
  – Application characterization (subsystem demand)
  – Empirical quantification of possible power savings (upper bound)
  – Proactive subsystem power management at system level
Application Characterization

Opportunities for low power modes

Need to be careful

HPL
Bonnie++
Beff_io

Component-based Power Management

• Application profiling
  – CPU utilization
  – Memory utilization (L2 cache misses is an indicator)
  – Disk utilization
  – Network traffic

• Proactive power management
  – Based on the application profile
  – Using component/subsystems low power modes
    • CPU DVFS
    • Memory (e.g., switching off banks)
    • Disk (spin down/up)
    • NIC (sleep modes)

Typical system power breakdown

CPU
Others
Memory
Network
Disk

Application-Aware Proactive Aggressive Power Management (PAPM)

Energy consumed in an interval of time $t_i$ for a subsystem $sys$:

$$E_{sys}^{t_i} = P_{active}^{sys} \cdot t_{i, active} + P_{idle}^{sys} \cdot t_{i, idle} + E_{on/off}$$

- $E_{on/off}$: Energy cost associate to a transition
- $t_{i, active}$: Time duration in active mode
- $t_{i, idle}$: Time duration in idle mode
- $P_{active}^{sys}$: Power if the subsystem $sys$ is active
- $P_{idle}^{sys}$: Power if the subsystem $sys$ is idle (low power mode)

where $sys = \{mem, disk, nic\}$ and $P = P_{static} + P_{dynamic}$

$$P_{dynamic} = P_{cpu} + P_{mem} + P_{disk} + P_{nic}$$

PAPM Algorithm

- **Idle-Condition** checks if the subsystem is going to be idle in the next time interval
- **Time-Condition** checks if it is feasible to transition to any of the available lower power states based on the latency of the subsystems and the idle time between the two active periods
- **Energy-Condition** checks if it is worthwhile to transition the system to a low power state and if any energy savings will be achieved
Subsystems Energy Contribution – Simulation

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>DVFS</th>
<th>Run Time (s)</th>
<th>Energy (J)</th>
<th>CPU</th>
<th>Memory</th>
<th>Disk</th>
<th>NIC</th>
<th>Total (J)</th>
<th>Energy Savings (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPL</td>
<td>×</td>
<td>1.382 s</td>
<td>298,540 J</td>
<td>-</td>
<td>1,380 J</td>
<td>5,338 J</td>
<td>240 J</td>
<td>6,958 J</td>
<td>2.33%</td>
</tr>
<tr>
<td>b_eff_io</td>
<td>√</td>
<td>1.385 s</td>
<td>292,824 J</td>
<td>-</td>
<td>1,380 J</td>
<td>5,722 J</td>
<td>2764 J</td>
<td>6,421 J</td>
<td>3.80%</td>
</tr>
<tr>
<td>bonnie++</td>
<td>×</td>
<td>1.247 s</td>
<td>190,613 J</td>
<td>-</td>
<td>5,297 J</td>
<td>3,263 J</td>
<td>574 J</td>
<td>3,841 J</td>
<td>4.63%</td>
</tr>
<tr>
<td>TailBench</td>
<td>√</td>
<td>1.13 s</td>
<td>244,473 J</td>
<td>-</td>
<td>3,377 J</td>
<td>7,297 J</td>
<td>1,979 J</td>
<td>20,084 J</td>
<td>5.97%</td>
</tr>
<tr>
<td>FFTW</td>
<td>×</td>
<td>1.052 s</td>
<td>198,621 J</td>
<td>-</td>
<td>2,112 J</td>
<td>6,927 J</td>
<td>297 J</td>
<td>9,336 J</td>
<td>4.74%</td>
</tr>
</tbody>
</table>

- Minimal performance penalty – 0.22% on an average
- Can improve the energy efficiency by up to ~8%, depending on the workload profile for these benchmarks (potentially more for other workloads)

Energy Savings – Experimental Validation

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Configuration</th>
<th>Run Time (s)</th>
<th>%</th>
<th>Energy (J)</th>
<th>%</th>
<th>EDP</th>
<th>%</th>
</tr>
</thead>
<tbody>
<tr>
<td>HPL</td>
<td>Reference</td>
<td>1.385 s</td>
<td>-</td>
<td>287,906 J</td>
<td>-</td>
<td>589,749 J</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>PAPM+SSD</td>
<td>1.385 s</td>
<td>+0.14%</td>
<td>281,559 J</td>
<td>-</td>
<td>589,059 J</td>
<td>-</td>
</tr>
<tr>
<td>b_eff_io</td>
<td>Reference</td>
<td>1.249 s</td>
<td>-</td>
<td>182,904 J</td>
<td>-</td>
<td>328,447 J</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>PAPM+SSD</td>
<td>1.249 s</td>
<td>+0.08%</td>
<td>168,606 J</td>
<td>-</td>
<td>328,100 J</td>
<td>-</td>
</tr>
<tr>
<td>bonnie++</td>
<td>Reference</td>
<td>1.137 s</td>
<td>-</td>
<td>229,446 J</td>
<td>-</td>
<td>457,291 J</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>PAPM+SSD</td>
<td>1.137 s</td>
<td>+0.08%</td>
<td>229,446 J</td>
<td>-</td>
<td>457,291 J</td>
<td>-</td>
</tr>
<tr>
<td>FFTW</td>
<td>Reference</td>
<td>1.055 s</td>
<td>-</td>
<td>187,677 J</td>
<td>-</td>
<td>374,589 J</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>PAPM+SSD</td>
<td>1.055 s</td>
<td>+0.19%</td>
<td>177,071 J</td>
<td>-</td>
<td>374,589 J</td>
<td>-</td>
</tr>
</tbody>
</table>

- Reference: State-of-the-art DVFS
- PAPM: Proactive Aggressive Power Management
- PAPM+SSD: PAPM with the use of SSD rather than disk
Applications exhibit heterogeneous BW requirements over time

**Approach:**
- Apply frequency and voltage scaling per channel
- Affinity to channels: more channels → more opportunities
- Controlling channels independently → more energy savings

Application-awareness provides higher energy conservation with low penalty in performance

---

**Use Case: Multi-channel DRAM Power Management [HiPC’11]**

- Applications exhibit heterogeneous BW requirements over time
- Approach:
  - Apply frequency and voltage scaling per channel
  - Affinity to channels: more channels → more opportunities
  - Controlling channels independently → more energy savings
- Application-awareness provides higher energy conservation with low penalty in performance

---

**Application-aware Power Management: Lessons Learned**

- Application-aware power control can lead to improved energy efficiency without significant performance penalty
- Power management at the subsystem level can not be neglected
- Low power devices and upcoming technologies (e.g., SSD, storage-level memory) can significantly increase energy efficiency
- However, power management at a single level is not enough
- A *cross-layer* approach is required!
2. Programming Support for GreenHPC

- Objective: Develop programming support to enable applications to provide hints to drive runtime power management

- Overall approach [HPDC12]
  - Language extensions ("hints") to drive/tune runtime power management
  - Runtime middleware for application-aware cross-layer power management
  - Implementation: PGAS applications on many-core platform (i.e., Intel SCC)
Architecture for PGAS Programming Support

**Application Stack**
- UPC application
- UPC runtime
  - gasnet
  - RCK-MPI
- Message Passing Buffer

**Power Management**
- PM application layer
- PM runtime layer
  - L2CFG unused bit
  - "Hints"

**Implementation using:**
- Berkeley UPC runtime
- Intel SCC platform

Experimental Platform

**Intel SCC platform**
- Many-core prototype by Intel Labs Tera-scale Program
- 48 x86 P54C Pentium cores
- 24-router on-die mesh network, hardware message-passing
- Per-core 16 KB L1 cache, 256 KB L2 cache
- Metering infrastructure
- Frequency (per 2 core groups) and voltage (per 8 core groups) scaling
- Power ranges from 25W to 125W, i.e., 0.5W-2.6W per core
Application in the Loop

- Application-awareness can improve runtime power management (in combination with system level power controls)
- Example: NAS FT profiling
  - Only certain operations are candidate for power management
    - Remote memory access (memget) and barrier (wait) operations
  - Both operations have lots of very-short calls and several med-long operations (short calls result in large overheads)

Language Extensions Provide User “hints”

- Analogous to processor DVFS governors at the OS level
  - PM PERFORMANCE, maximum performance
    - full power
  - PM CONSERVATIVE, balance power/performance
    - low power modes during slack
  - PM POWER, minimum power, limited delay penalty
    - low voltage level, regardless application’s slack periods
  - PM AGGRESSIVE POWER, maximum energy save
    - lowest voltage and frequency levels
Use Case 1: NAS FT – UPC implementation

- Runtime power management policy selects appropriate tradeoff between energy and delay
  - Larger energy savings -> higher time penalties
- Energy reduction with little time penalty is possible
  - PM CONSERVATIVE 7% results in energy savings with a 0.4% time penalty
- Language extensions provide hints to guide runtime policy selection
  - Other hints can obtain higher energy savings (45%) with a higher time penalty cost

Use Case 2: Sobel Filter Application

- Sobel filter has two different phases
- Two studied strategies
  - Using the same policy during the whole application execution
  - Using hints during the initial phase (e.g., PM_POWER)
- Hints avoid time delay and maintains energy savings
  - The right power mode cannot be identified at runtime or system level

<table>
<thead>
<tr>
<th>Language extension (user hints)</th>
<th>Sobel - fix policy</th>
<th>Sobel - dynamic policy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time (%)</td>
<td>Energy (%)</td>
</tr>
<tr>
<td>PM_PERFORMANCE</td>
<td>100.0</td>
<td>100.0</td>
</tr>
<tr>
<td>PM_POWER</td>
<td>147.6</td>
<td>75.0</td>
</tr>
<tr>
<td>PM_AGGRESSIVE_POWER</td>
<td>193.8</td>
<td>72.8</td>
</tr>
<tr>
<td>PM_CONSERVATIVE</td>
<td>118.0</td>
<td>73.7</td>
</tr>
</tbody>
</table>

Power over time (Sobel)
Programming Support for GreenHPC: Lessons Learned

- Cross-layer power management enables a wider range of energy and performance behaviors
- Application-awareness improves runtime power management
- Language extensions (via user-level hints) enable a better selection of the appropriate energy/performance tradeoff
- Cross-layer decisions can help address hardware power management limitations
  - E.g., Intel SCC provides per-tile frequency scaling and per-voltage domain voltage scaling

3. Power-aware data management
Scientific Discovery through Simulations - III

- Scientific simulations running on high-end computing systems generate huge amounts of data!
  - If a single core produces 2MB/minute on average, one of these machines could generate simulation data between $\sim 170$TB per hour -> $\sim 700$PB per day -> $\sim 1.4$EB per year

- Successful scientific discovery depends on a comprehensive understanding of this enormous simulation data

How we enable the computation scientists to efficiently manage and explore extreme scale data: “find the needles in haystack”??

Traditional Simulation -> Insight Pipelines Break Down

- Traditional simulation -> insight pipeline:
  - Run large-scale simulations on large supercomputers
  - Dump data on parallel disk systems
  - Export data to archives
  - Move data to users’ sites – usually selected subsets
  - Perform data manipulations and analysis on mid-size clusters
  - Collect experimental / observational data
  - Move to analysis sites
  - Perform comparison of experimental/observational to validate simulation data
Challenges Faced by Traditional HPC Data Pipelines

- **Data analysis challenge**
  - Can current data mining, manipulation and visualization algorithms still work effectively on extreme scale machine?

- **I/O challenge**
  - Increasing performance gap: disks are outpaced by computing speed

- **Data movement challenge**
  - Lots of data movement between simulation and analysis machines, between coupled multi-physics simulation components → longer latencies
  - Improving data locality is critical: do work where the data resides!

- **Energy challenge**
  - Future extreme systems are designed to have low-power chips – however, much greater power consumption will be due to memory and data movement!

*The costs of data movement are increasing and dominating!*

---

The Cost of Data Movement

- Moving data between node memory and persistent storage is slow!
- The energy cost of moving data is a significant concern

\[
\text{Energy}_{\text{move\_data}} = \frac{\text{bitrate} \times \text{length}^2}{\text{cross\_section\_area\_of\_wire}}
\]

Rethinking the Data Management Pipeline – Hybrid Staging + In-Situ & In-Transit Execution

- Exploit multi-levels in-memory Hybrid Data Staging to:
  - Decrease the gap between CPU and IO speeds
  - Dynamically deploy and execute data analytical or pre-processing operations either in-situ or in-transit
  - Improve IO write performance

Data Staging over Deep Memory Hierarchy

Motivation
- Small DRAM capacity per core – even aggregated memory on dedicated nodes can hardly keep all coupled data (given the ratio of resource allocations for compute nodes and dedicated nodes)

Hybrid Staging
- Spans horizontally across compute nodes of both primary and secondary resources
- Spans vertically across the multi-level memory hierarchy, e.g. DRAM/NVRAM/SSD, to extend the capacity of in-memory data staging
Deep Memory Hierarchy

- NVRAM bridges latency/performance gap between DRAM & disk
- NVRAM usage modes
  - I/O staging and data "caching"
  - Out-of-core computations (virtual memory)
  - "In-the-space" data manipulation and/or reduction
- Fusion-io technology (ioDrive)

Energy-Performance Tradeoffs in the Data Analytics Pipeline

Characterize power/performance behaviors of deep memory hierarchy using benchmarks

**Lessons learned:** Storage-class NVRAM memory is a good candidate as intermediate storage
Energy-Performance Tradeoffs in the Analytics Pipeline

Impact of data placement and movement
- Canonical system architecture composed of DRAM, NVRAM, SSD and disk storage levels and networked staging area
- Canonical workflow architecture that can support in-situ and in-transit analytics
- **Lessons learned**: Data placement/movement patterns significantly impact performance and energy

Role of the quality of solution
- **Lessons learned**: Frequency of analytics is driven by the dynamics of features of interest, but is limited by node architecture.

Power Behavior of In-situ Analytics Pipeline

- **Motivation**: Coupled simulation workflows use online data processing to reduce data movement. Need to explore energy/performance tradeoffs.
- **Use case**: Combustion simulation workflow with an in-situ data analytics pipeline.

- **Energy model** based on machine-independent application profile
  \[
  E = E_{\text{system}} + E_{\text{comm}}.
  \]
  \[
  E_{\text{system}} = T \cdot (p_{\text{static}} + p_{\text{dynamic}}),
  \]
  \[
  E_{\text{comm}} = \sum_{i=1}^{n} \left( \frac{\text{data}_{i}}{\text{BW}_{\text{comm}}} \right) \cdot (p_{\text{comm}} + (P_{\text{comm}} + P_{\text{comm2}})).
  \]

- **Model validation** on small cluster

![Power Behavior of In-situ Analytics Pipeline](image)
Power behavior of the in-situ analytics pipeline

- Study various co-design trade-offs
  - Algorithm design choices
  - Runtime application deployment
  - System architecture

- Algorithm parameter impact is quantified

Topologies analysis: parallel merge tree algorithm

Power Behavior of In-situ Analytics Pipeline

- Communication patterns depend on:
  - Algorithmic configuration (Fan-in)
  - MPI processes mapping (ranks/cpu)
  - System architecture (e.g., 16 cores/node vs. 32 cores/node)

- Data motion energy depends on transfers

Lessons learned: Energy for data motion is impacted by algorithm and system
Conclusions

- Energy-related challenges have become dominating concerns for scientific applications at extreme scales
- Costs (energy, latency) related to transporting, processing and analyzing increasing data volumes and rates are limiting the insights from extreme scale applications
- Multiple orders of magnitude power/energy reduction are required – a cross-layer and application-aware strategy is necessary
  - Innovations necessary at multiple levels (algorithms, programming, runtime, OS, etc.)
- Energy/power-efficiency in combination with other objectives – understanding tradeoffs are important
  - Quality of solution, Performance, Reliability, etc.
- Co-design is essential

Not All Improvements Come from HW/System

---

Magnetic Fusion Energy: "Effective speed" increases came from both faster hardware and improved algorithms

Effective sustained speed in equivalent gigaflops

- Energetic Simulator (Japan)
- Improved electron models
- Improved linear solvers
- Improved semi-implicit
- Effective speed from hardware improvements alone

Calendar Year


Ack. David Keyes
Thank You!

Manish Parashar, Ph.D.
Prof., Dept. of Electrical & Computer Engr.
Rutgers Discovery Informatics Institute (RDI²)
Cloud & Autonomic Computing Center (CAC)
Rutgers, The State University of New Jersey

Email: parashar@rutgers.edu
WWW: rdi2.rutgers.edu