Porting of GEOPM to IBM Power8 with NVLink microarchitecture

Vadim Elisseev\textsuperscript{1}  Miloš Puzović\textsuperscript{2}

\textsuperscript{1}IBM Research

\textsuperscript{2}Hartree Centre

BoF: PowerAPI, GEOPM and Redfish: Open Interfaces for Power/Energy Measurement and Control
The performance of future large-scale HPC and data-centric systems will be constrained by power costs

- Optimizing performance under power constraints
  - Reduce power consumption of idle nodes
  - Reduce power consumption of active nodes

\[ E(t) = \int_0^t P(x) \, dx \]

Reduce and Control Energy
- Reduce Power
- Reduce Time
- Reduce Power and Time
- Deal with Power Variations

EAS Policies
- Minimize Time to Solution
- Minimize Power to Solution
- Power Capping
- Energy Budget
Motivation

Implementing EAS

Model Input:
- Performance counters
- Power energy consumption
- Historical Jobs Data
- Telemetry Data

Model Requirements:
- Support wide range of workloads
- Work with various schedulers

Cluster/job level EAS policies:
- Min time to solution
- Min energy to solution
- Energy budget
- Idle nodes
- Energy aware fair share

Job level EAS policies:
- Min time to solution
- Min energy to solution
- Energy budget
- Power capping
- Load balancing
Why **Global Extensible Open Power Manager**?

- **Common goals with our work on EAS:**
  1. unlock more performance in power-limited systems
  2. accelerate innovation in power management
  3. enable researches to focus effort on algorithms
  4. drive codesign of power and performance management features in new processors

- **Current functionality aligns with our aims too:**
  - Framework easily pluggable to the existing data-center and HPC schedulers
  - Scaling challenge for future exascale machines addressed successfully via tree-hierarchical design and hierarchical policies
  - Leverage application-awareness and learning to recognise patterns to optimise decisions
  - Comprehend and mitigate dynamic load imbalances
  - Extensibility provided via plugins
Progress of the port to date

- Port of GEOPM *observation* part on IBM Power\(^1\) with **five** new classes:
  - OCCPlatform extends Platform to describe RAPL-like observers and controllers on Power
  - NVLMPPlatform extends Platform to describe RAPL-like observers and controllers on GPU
  - Power8NVLinkPlatformImp extends PlatformImp to implement sampling and management by referring work to following two classes and aggregating results from them:
    - PowerPlatformImp extends PlatformImp for information from CPUs
    - PascalPlatformImp extends PlatformImp for information from GPUs

- To collect power and performance data we are using:
  - on CPU:
    - libpfm4 library for performance counters
    - sensor data from */sys/devices/system/cpu/occ_sensors* for power consumption
  - on GPU NVIDIA Management Library (NVML), a C-based API for monitoring and managing

- Port of the GEOPM framework to IBM Power8 with NVLink builds on:
  - IBM C/C++ and FORTRAN Compiler and IBM Spectrum MPI
  - GNU toolchain and OpenMPI 2.1.2

\(^1\)Power8 dual-socket CPU + Nvidia Pascal P100 GPU with high-speed embedded, proprietary and private NVLink interface
Example results

- **Setup:**
  - 2× nodes - dual-socket 160 threaded cores IBM Power 8 CPU, 4× Nvidia Pascal P100 GPUs and 512GB RAM
  - GCC version 4.8.5 and OpenMPI version 2.1.1
  - Workload: DL__MESO - *mesoscale simulation package*, within Top 3 of all workloads (32.73% share) @ Hartree
  - Plot (left): “geopmplotter -vp combined_power -smooth 15 .” (bgeltz-plotter-runtime branch)

- **Further exploration:**
  - understand where discrepancies in power across nodes are coming
  - scale the running from 2 nodes to 30+ nodes
  - run the GPU-optimised version of DL__MESO
The future immediate work

- Remove workarounds where we are using #defines to separate between x86 and Power:
  - in Controller.cpp plugin_desc.platform only knows of RAPL platform
  - avoid using cpuid.h to figure out target (non-existent for Power)
  - more generic implementation of CRC32 (do not rely on x86 instructions)
  - is_updated() to find out whether fresh performance/power data is available relies on RAPL

- Implement *management* part of the GEOPM framework on Power microarchitecture
  - this is already on-going work,
  - the most difficult part is being able to do DVFS-like modifications in user mode

- Integration with:
  - High Performance Computing batch scheduler such as LSF, and
  - manager of containerized applications such as Mesos and/or Kubernetes.

- Use these extensions to GEOPM to identify imbalance in applications ran on Hartree machines and optimise them for a certain objective function:
  - Widely used applications such as CodeSaturne and DL_MESO
  - Demonstrate speed up and energy savings