ECP PowerSteering: IBM GEOPM port

Introduction

The following document provides an in-depth overview of the IBM GEOPM port developed as part of ECP PowerSteering effort.

Build Environment

Dependencies:

- All base GEOPM dependencies
- OPAL firmware update with power capping exports.
- OPAL firmware with OCC sensors for power, energy, frequency measurements
- Performance monitoring events (IOCTL offset 319) enabled with root access

Build instructions:

To configure the GEOPM repository on a vanilla IBM P9 system, use the following command:

```
./configure \
      --prefix=<location for binary installation>/install \
      CPPFLAGS=-I/usr/local/cuda/include/ \
      CFLAGS=-I/usr/local/cuda/include/ \
      LDFLAGS=-L//usr/lib64/nvidia/ \
      LIBS=-lnvidia-ml
```

To build, use

```
make
make install
```

Important new files added:

- OCCPlatform.cpp: OCC Platform plugin hook-up to GEOPM policy infrastructure.
- OCCPlatform.hpp: OCC Platform plugin class and method declarations.
- PowerPlatformImp.cpp: IBM P9 platform power and performance telemetry and control implementation
- PowerPlatformImp.hpp: IBM P9 platform class and method declarations.
- Other headers defining architecture and performance event registers.

Existing source files with major changes:

- PlatformImp.cpp:
  CPU architecture file descriptor

Performance event enumeration and initialization
Performance event telemetry initialization

- PlatformImp.hpp:
  Platform ID registration
  Platform override method declaration

Source files with minor changes:

- Controller.cpp: Platform plugin name registration
- GlobalPolicy.cpp: Registration of platform plugin in the GEOPM base policy registration
- PlatformFactory.cpp: Updated factory class with IBM Power9 platform ID and plugin registration
- geopm_hash.h: CRC32 hash function declaration
- geopm_arch.h: Platform ID definition

Detailed description of the new functionality:

- Power9 platform power and energy implementation
  - Define bounds for node level power limits
  - Throttle limit for DVFS
  - Telemetry signal initialization
  - Individual and batch signal read for the following telemetry data:
    - CPU power usage
    - CPU energy usage
    - CPU frequency
    - GPU power usage
    - Memory power usage
    - # Instructions retired
    - # Unhalted clock core cycles
    - Memory read bandwidth
  - Methods introductions:
    -
  - File descriptors used
    - Power9 telemetry: /sys/firmware/opal/exports/occ_inband_sensors
    - Power9 frequency measurements: /sys/devices/system/cpu/cpufreq/policy*/scaling_cur_freq
    - Power9 DVFS control: /sys/devices/system/cpu/cpufreq/policy*/scaling_setspeed
    - Power9 node power capping: /sys/firmware/opal/powercap/system-powercap/powercap-current
    - Power9 node power limits:
      - /sys/firmware/opal/powercap/system-powercap/powercap-min

        - /sys/firmware/opal/powercap/system-powercap/powercap-max

    - H/w counter map to inform GEOPM tracing functionality about the available counters.

- OCC Platform plugin interface
    - Platform plugin initialization
    - Number of power management domains
    - Operating bounds on node power
    - Enable application telemetry sample generation

Evaluation Platform Specification

IBM Power9 - "Witherspoon"
CPU ID: PowerNV 8335-GTH, 2.2 (pvr 004e 1202)
Number of cores: 160 4-way SMT
System memory: 66 GB
GPU: Tesla V100-SXM2 (Nvidia)
Operating System: Red Hat Linux
Compiler toolchain: GNU C/C++, GNU Fortran
Other software libraries: MPICH2 with Fortran support, Ruby Gem with Ronn, GNU Make, doxygen, math libraries.

DVFS-based CPU power model

Since the OPAL-based firmware power interface is under development, we use DVFS as the primary method for power enforcement on the IBM Power9 platform. DVFS control is provided through the following system interface:

```
/sys/devices/system/cpu/cpufreq/policy[0-156]/scaling_setspeed
```

The frequency scaling (policy*) interface is provided in steps of 4 starting at 0 such that DVFS can be enforced at core level with 4-way SMT. Since the base Intel GEOPM functionality does not employ core-level DVFS control for power enforcement, we limit the DVFS control for this port at the CPU level by assigning equal DVFS levels to all cores on the same CPU. The available range of frequencies on our "Witherspoon" system is as follows:

```
cat /sys/devices/system/cpu/cpufreq/policy[0-
156]/scaling_available_frequencies

3000000 2983000 2966000 2950000 2933000 2916000 2900000 2883000
2866000 2850000 2833000 2816000 2800000 2783000 2766000 2750000
2733000 2716000 2700000 2683000 2666000 2650000 2633000 2616000
2600000 2583000 2566000 2550000 2533000 2516000 2500000 2483000
2466000 2450000 2433000 2416000 2400000 2383000 2366000 2350000
2333000 2316000 2300000
```

We have developed a power model based on the above range of operating frequencies on the IBM P9 platform. The following describes the model in brief:

$$P_{CPU} = \alpha.F + C$$

where, P_{CPU} : P9 CPU power usage (watts),
F : CPU frequency (GHz)
α : Coefficient of frequency scaling
C : Constant offset for base frequency <-> correlation.

We developed this model based on empirical power usage measurements at all frequency steps for a pre-determined set of benchmarks. By applying linear regression on the empirical data, we determined the values for α and C as follows:

α = 0.0478
C = 0.00145

In case of inaccurate prediction of target power usage at a selected frequency step, our algorithm adjusts the frequency step to the closest target power usage specified in the policy assignment in the subsequent iteration (walk-down) of the decider algorithm.

Limitations

We list the limitations of the current port of the GEOPM functionality. The limitations listed here are primarily due to the lack of power-management functionality that must be provided by the OEM (in this case, our collaborators at IBM).

1. Support for complete node power limit in GEOPM's power governing logic.
2. Support for GPU-specific power limit in GEOPM's power governing logic.
3. Missing support in GEOPM for core-aware DVFS-based power model.
4. At-scale testing of GEOPM's power balancing logic.

Future work

As part of ongoing and future work, we plan to make the following enhancements to the existing functionality. Note that specific firmware/platform features on the IBM P9 platform need to be enabled in order to make progress on the enhancements listed below.

- Add functionality to include node-level power limit control into the governing capability of GEOPM.
- Make the power-governing logic of GEOPM at the node level aware of the GPU-specific power limit controls. This effort needs additional support from the BMC firmware which

manages proportional power assignment to all GPUs on the system. This specific effort may involve disproportionate power assignment to individual GPU units depending on the load imbalance in the GPU portion of the application/workflow.

- We plan to test and improve the power-balancing logic of GEOPM once a large-scale IBM Power9 system is available with the aforementioned firmware/platform features enabled by the OEM.

References

- Part of this work was derived based on our collaborative effort with IBM: https://github.com/milpuz/geopm
- Reading IBM P9 inband sensors: https://github.com/shilpasri/inband_sensors
- OCC: https://github.com/open-power/occ
- IBM P9 user manual and complete performance event reference.