DPG-CESG-ASE-Power

GEOPM Service:

Application Configuration of Hardware Settings

Christopher Cantalupo <christopher.m.cantalupo@intel.com>

intel®

Modern Heterogeneous Systems

Increasing System Complexity

- New Intel CPU features: SST, HWP, RAPL, Memory bandwidth monitoring, and cache monitoring
- Accelerators: GPGPU, AI and FPGA
- Memory: on package HBM, persistent high-capacity, CXL bus
- Advanced packaging techniques and modular CPU IP

Static Configs are Limiting

- Data center hosts different apps on the same platform with time slices
- One-size-fits-all hardware configurations are inadequate
- Fail to address customer needs for performance, energy efficiency, and customization
- Hardware features not effectively utilized or disabled by customers
- Performance prioritization for all use cases leads to energy inefficiencies

Security Model

Application Isolation w/o GEOPM

- Hardware settings are traditionally privileged
  - Persistent changes to hardware settings impact performance for all users
- Isolation of hardware between applications
  - HPC: Resource manager (SLURM, PBSPro) with single tenancy
  - Cloud: Linux cgroup cpuset for VM and container isolation
  - Cloud: Bare metal offerings

GEOPM Service Access Model

- Save all settings before enabling one controlling process at-a-time
- Track controller process ID and restore settings when process ends
- Provide a name-based access management interface for system administrators
- Administrators use names to query information and configure user access
- Combine with cgroup restrictions at runtime
- Default access controls for users, extensible with Unix groups

Control System for Heterogeneous Platforms

Arbitrary Device Support

- Support for read and write access
- IOGroup plugin for each device
- Read / write – single or batched
- Save / restore all writable settings
- Topology maps devices to CPUs

Access Management

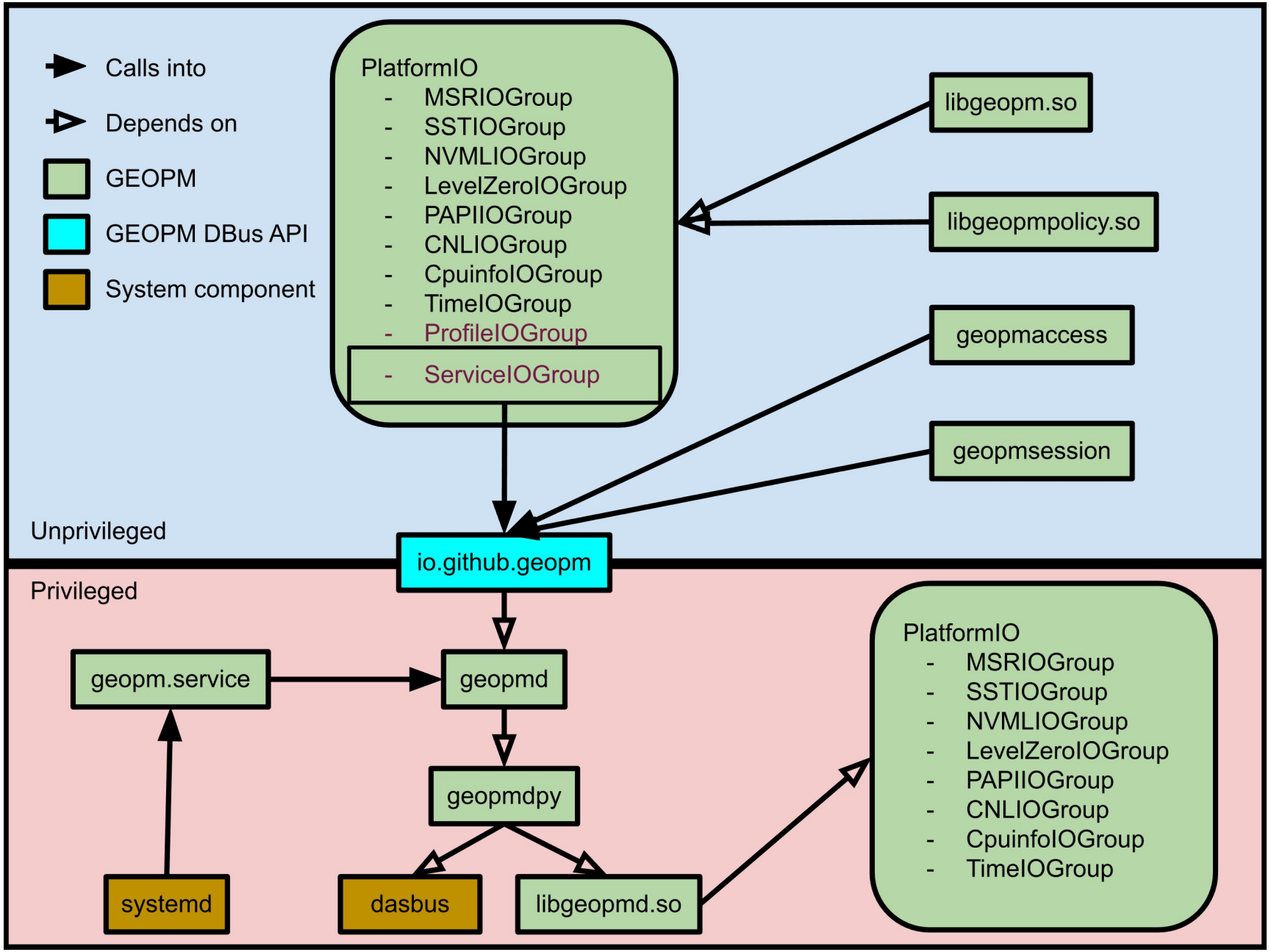
- Read “signals” and write “controls”
- Signals & controls have string IDs
- Access list is maintained by admin
- Descriptions guide administrator
- Default access list for all users
- Extend access by Unix group
- Restrict by cgroup cpuset

Concepts and Terms

- PlatformIO – Platform abstraction with CLI, and bindings for C, C++, and Python
- Signals – Named value with domain that can be read in SI units (or unitless)
- Controls – Named value with domain that can be written in SI units (or unitless)
- Domain – A platform component that is associated with one or more CPUs
- Allowed List – List of signals or controls configured for user access
- Session – Active communication channel between geompd and a client
- Mode – Session read/write attribute, starts as “r” may be converted “rw”
- Batch server – Process forked by geopmd to support batch access for one client
- DBus – Systemd interface for geopmd / client communication

GEOPM Service Features

- Fine Grained Access Management
  - System administrator can configure the signals and controls that each user/group has access to
  - Respect Linux cgroup(7) cpuset restrictions for each client process request
- Save / Restore of Controls
  - Do not allow user to make persistent changes to hardware settings
  - Track each client PID, and limit lifetime of client settings to the lifetime of the client process
- High Performance Interface
  - Provide a fast-path for batch access to signals and controls through inter-process shared memory
  - Use DBus to validate access to set of signals and controls before opening shared memory window
  - Shared memory and POSIX signals enable fast client access after permissions have been established
- Easy Extensibility
  - The IOGroup defines interfaces required for extension and enables shared object plugins
  - Signal and control names can be overridden by IOGroups and preference is determined by plugin load order
  - Support exists for MSR, SST, NVML, (LevelZero and PAPI available from feature branches)


Calls into
Depends on
GEOPM
GEOPM DBus API
System component
PlatformIO
- MSRIOGroup
- SSTIOGroup
- NVMLIOGroup
- LevelZeroIOGroup
- PAPIIOGroup
- CNLIOGroup
- CpuinfoIOGroup
- TimeIOGroup
- ProfileIOGroup
- ServiceIOGroup
libgeopm.so
libgeopmpolicy.so
geopmaccess
geopmsession
Unprivileged
io.github.geopm
Privileged
geopm.service
geopmd
geopmdpy
systemd
dasbus
libgeopmd.so
PlatformIO
- MSRIOGroup
- SSTIOGroup
- NVMLIOGroup
- LevelZeroIOGroup
- PAPIIOGroup
- CNLIOGroup
- CpuinfoIOGroup
- TimeIOGroup

Example: Limit CPU Frequency

https://github.com/geopm/geopm/blob/geopm-service/service/test/test_write_session.sh

Writing to 8 bits of the IA32_PERF_CTL Register

```
# PARAMETERS
CONTROL=MSR::PERF_CTL:FREQ
DOMAIN=core
DOMAIN_IDX=0
SETTING=2000000000.0
REQUEST="${CONTROL} ${DOMAIN} ${DOMAIN_IDX}"

# READ START VALUE OF CONTROL REGISTER
START_VALUE=$(echo ${REQUEST} | geopmsession)

# START A SESSION THAT WRITES THE CONTROL VALUE
echo "${REQUEST} ${SETTING}" | geopmsession -w -t 10 &
SESSION_ID=$!
sleep 1

# READ THE CONTROLLED REGISTER
SESSION_VALUE=$(echo ${REQUEST} | geopmsession)

# END THE SESSION
kill -9 ${SESSION_ID}
sleep 1

# READ THE RESTORED REGISTER
END_VALUE=$(echo ${REQUEST} | geopmsession)
```

```
# ERROR FUNCTION
test_error() {
    echo "Error: $1" 1>&2
    exit -1
}

# CHECK THAT IT IS DIFFERENT THAN THE TEST VALUE
test ${SETTING} != ${START_VALUE} ||
    test_error "Start value for the control equals test value"

# CHECK THAT THE REGISTER WAS CHANGED DURING THE SESSION
test ${SETTING} == ${SESSION_VALUE} ||
    test_error "Control is not set during the session"

# CHECK THAT SAVE/RESTORE WORKED
test ${START_VALUE} == ${END_VALUE} ||
    test_error "Control is not restored after the session"
```

Example Walk Through

- **`geopmsession`** CLI uses GEOPM service to open session for reading or writing
- User sets the maximum core frequency using the IA32_PERF_CTL register
- **`"MSR::PERF_CTL::FREQ"`**
  - The GEOPM service signal / control name
  - Represents the 8 bits of the register that control maximum core frequency
  - **`"CPU_FREQUENCY"`** is a control alias
- All signals and controls are in SI units, so in this case Hz

- Permissions requirements
  - Proper execution of the script requires user access for **`"MSR::PERF_CTL::FREQ"`**
  - This name must be in an allowed signal list and an allowed control list (default or group)
  - Only access to those 8 bits is granted by the name appearing in the lists
  - The user process must have all CPUs on core zero in their cgroup cpuset
- The test shows that the user can read, and write to the register
- The value is restored by the service after the session is killed

Example: Give Access to Frequency Limit

https://github.com/geopm/geopm/blob/geopm-service/service/test/test_su_give_access.sh

Enable writing to 8 bits of the IA32_PERF_CTL Register

```
# PARAMETERS
CONTROL=MSR::PERF_CTL:FREQ
TEST_USER=test

# SAVE INITIAL ACCESS SETTINGS
SAVE_SIGNALS=$(mktemp)
SAVE_CONTROLS=$(mktemp)
geopmaccess > ${SAVE_SIGNALS}
geopmaccess -c > ${SAVE_CONTROLS}

# REMOVE CONTROL FROM ACCESS LIST FOR READING AND WRITING
grep -v ${CONTROL} ${SAVE_SIGNALS} > ${TEST_SIGNALS}
grep -v ${CONTROL} ${SAVE_CONTROLS} > ${TEST_CONTROLS}
geopmaccess -w < ${TEST_SIGNALS}
geopmaccess -w -c < ${TEST_CONTROLS}

# RUN WRITE SESSION TEST AND MAKE SURE IT FAILS
su ${TEST_USER} ${TEST_SCRIPT} &&
    test_error "Access to $CONTROL was disabled, but succeeded"
```

```
# ADD THE CONTROL INTO ACCESS LIST FOR READING AND WRITING
TEST_SIGNALS=$(mktemp)
TEST_CONTROLS=$(mktemp)
echo ${CONTROL} >> ${TEST_SIGNALS}
echo ${CONTROL} >> ${TEST_CONTROLS}
geopmaccess -w < ${TEST_SIGNALS}
geopmaccess -w -c < ${TEST_CONTROLS}

# RUN WRITE SESSION TEST AND MAKE SURE IT PASSES
su ${TEST_USER} ${TEST_SCRIPT} ||
    test_error "Access to $CONTROL was enabled, but failed"

# RESTORE INITIAL ACCESS SETTINGS
geopmaccess -w < ${SAVE_SIGNALS}
geopmaccess -w -c < ${SAVE_CONTROLS}
```

Example Walk Through

- `geopmaccess` is the main administrative CLI for the service
- This test is run as the root user
- The test first saves the existing access list
- Frequency control bits of the IA32_PERF_CTL register are removed from the default access list as root
- The `"$TEST_SCRIPT"` refers to the first example
- The test tries to run the script in the previous example as non-root
- The script will fail due to lack of permissions
- Access to the required bits is enabled by the root user
- The same test script is run again as an unprivileged user, but is expected to pass
- The starting access list is then restored

Dynamic Hardware Configuration with GEOPM

- Provides a safe mechanism for applications to modify hardware settings
  - Applications are enabled to actively modify hardware settings to adjust to application demands
  - Customers can take advantage of hardware settings that may provide special benefits for their use case
- Isolation of changes to application lifetime
  - Hardware settings are always reverted after the application ends
  - In many environments there are existing mechanisms to isolate applications to their own hardware while active
- Provides both a secure mechanism for verification and a fast path for efficiency
  - Many algorithms require low latency dynamic controls, and this speed is enabled by the batch interface
  - All access mechanisms provide a secure validation of user permissions, even the low latency interface
- IOGroup plugins provide a mechanism for expanding to new hardware features
  - Implementations can be dynamically loaded at runtime through an .so plugin infrastructure
  - All requirements for extending the service are defined by the IOGroup interface

GEOPM Github Resources

- Home page

  https://geopm.github.io/
- Source code main repository

  https://github.com/geopm/geopm
- Service feature branch README

  https://github.com/geopm/geopm/tree/geopm-service/service#readme
- Example scripts described in presentation

  https://github.com/geopm/geopm/tree/geopm-service/service/test
- Service pull request, posted comments or questions are appreciated

  https://github.com/geopm/geopm/pull/1628


intel