Dynamic Voltage and Frequency Scaling in Graphite

DVFS features NOT Released Yet
Dynamic Voltage and Frequency Scaling

- Commonly used power management technique
  - Also known as CPU throttling
- Save power by simultaneously decreasing voltage and frequency:
  \[ P = C V^2 f \]
  - Cubic dynamic power savings!
- Limitations
  - Changing frequency is fast but changing voltage is slow
  - Static power limits energy savings: frequency scaling alone may not reduce power and actually increases energy consumption
DVFS trends

• Off-chip voltage regulator
  – Most common DVFS implementation
  – Slow voltage transition (ms scale)
  – Limited number of adjustable domains

• On-chip voltage regulator
  – Intel Haswell architecture
    • 100x faster
    • 50x smaller

• Per tile DVFS
  – Independent voltage domain per tile
  – Voltage changes at nanosecond scale

• Multiple domains per tile
  – Fine grained DVFS control
  – Core/Caches/Network may run at different speeds and voltages
DVFS in Graphite

• Capabilities
  – Multiple levels of granularity:
    (1) System-wide; (2) per-tile; and (3) within tile voltage domains
  – DVFS modules:
    CORE, L1_ICACHE, L1_DCACHE, L2_CACHE, DIRECTORY, NETWORK
  – Graphite lets the user define arbitrary aggregate DVFS domains
  – Customizable voltage-frequency curves

• Modeled costs:
  – Asynchronous boundaries (modeled)
  – Voltage and frequency change (currently not modeled)
Defining DVFS domains

- **DVFS parameters are defined in** `carbon_sim.cfg`:
  - `[dvfs]
    
dvfs_domains = <init_freq, module1, module2, ...>, <...>, ...
    
    async_boundary_cost = 2

- **Single domain (default):**
  
dvfsDomains = <1.0, CORE, L1_ICACHE, L1_DCACHE, L2_CACHE, DIRECTORY, NETWORK_MEMORY, NETWORK_USER>

- **Two domains:**
  
dvfs_domains = <2.0, CORE, L1_ICACHE, L1_D_CACHE>, <1.0, L2_CACHE, DIRECTORY, NETWORK_MEMORY, NETWORK_USER>

- **Three domains:**
  
dvfs_domains = <1.0, CORE, L1_ICACHE, L1_D_CACHE>, <2.0, L2_CACHE, DIRECTORY>, <1.5, NETWORK_MEMORY, NETWORK_USER>
#include <carbon_user.h> // see file dvfs.h for all dvfs API functions

// query dvfs domain
module_t domain = getDVFSDomain(CORE);

// get frequency and voltage
double frequency, voltage;
int rc;
tile_id_t tile_id = 5;
rc = CarbonGetDVFS(tile_id, domain, &frequency, &voltage);

// set frequency and voltage
frequency = 2.0;
rc = CarbonSetDVFS(tile_id, domain, &frequency, AUTO);
// AUTO: select the best voltage for this frequency
// HOLD: change frequency while maintaining voltage

// change all tiles simultaneously
rc = CarbonSetDVFSAllTiles(domain, &frequency, AUTO);

// other API functions
rc = CarbonGetFrequency(tile_id, domain, &frequency);
rc = CarbonGetVoltage(tile_id, domain, &voltage);
Technology Files

(technology/dvfs_levels_45nm.cfg)

# All comment lines should start with the '#' character
# Please enter the DVFS levels in a table where each row
# has the following format:
# <Voltage (in V)> <Max-Frequency-Factor>
# Max-Frequency at a particular Voltage = Max-Frequency-Factor * Max-Frequency
1.1  1.0
1.06 0.88
1.02 0.77
0.98 0.65
0.94 0.54
0.9  0.42

• carbon_sim.cfg: max_frequency = 2.0
Return codes

• **Defined in file** `common/user/dvfs.h`

```c
/*
Return codes for all DVFS API functions:
0 : success
-1 : invalid tile id error
-2 : invalid module type or domain error
-3 : invalid voltage option error
-4 : invalid frequency error
-5 : above max frequency for current voltage error
*/
```

• **Note:** DVFS cannot be changed independently for the network
  - Use `CarbonSetDVFSAllTiles()` for the network