TECHNISCHE UNIVERSITÄT DRESDEN

Integrating Performance Analysis and Energy Efficiency Optimizations in a Unified Environment

EnA-HPC 2013

Robert Schöne (robert.schoene@tu-dresden.de)

Daniel Molka (daniel.molka@tu-dresden.de)

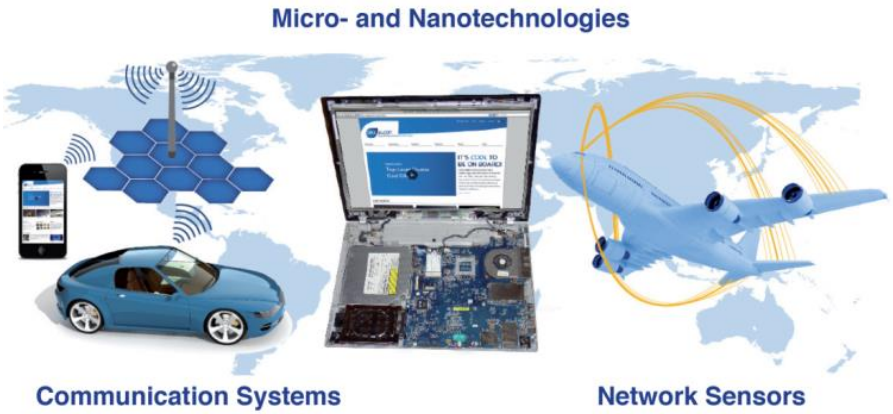
ZIH
Center for Information Services & High Performance Computing

Acknowledgement: Cool Silicon Spitzencluster

Sustainability through energy efficiency. The region "Silicon Saxony" acted on this maxim and founded the Leading Edge Cluster "Cool Silicon - Energy Efficiency Innovations from Silicon Saxony".

The Leading Edge Cluster Cool Silicon

- is aiming at providing a technological basis for increasing the energy efficiency in the information and communications technology sector.
- is assuring Germany and Europe a leading role in the key technology of "energy efficiency in the information technology sector".

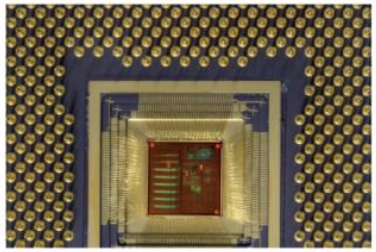


Helping to solve a global conflict: The worldwide desire for free communication collides with global climate protection targets, in case the energy consumption of communications technology keeps expanding at its current pace.

Cool Silicon - Areas

Area 1
Micro- und Nanotechnologies

The core objective of the Area 1 project partners is the development of basis technologies, analysis and production methods for the production of energy efficient electronics and their application in order to decrease the energy consumption of computer systems.



Area 2
Communication Systems

In Area 2, the research and development projects are focussing on the improvement of energy efficiency in communications infrastructures and mobile devices.

Area 3
Network Sensors

The project CoolSensorNet is the Lead Project of Area 3. It conducts research on the whole electronic chain's specific requirements, including sensors, analog electronics, A-D converters, processor systems and the telemetry unit.

Funded via BMBF 13N10186

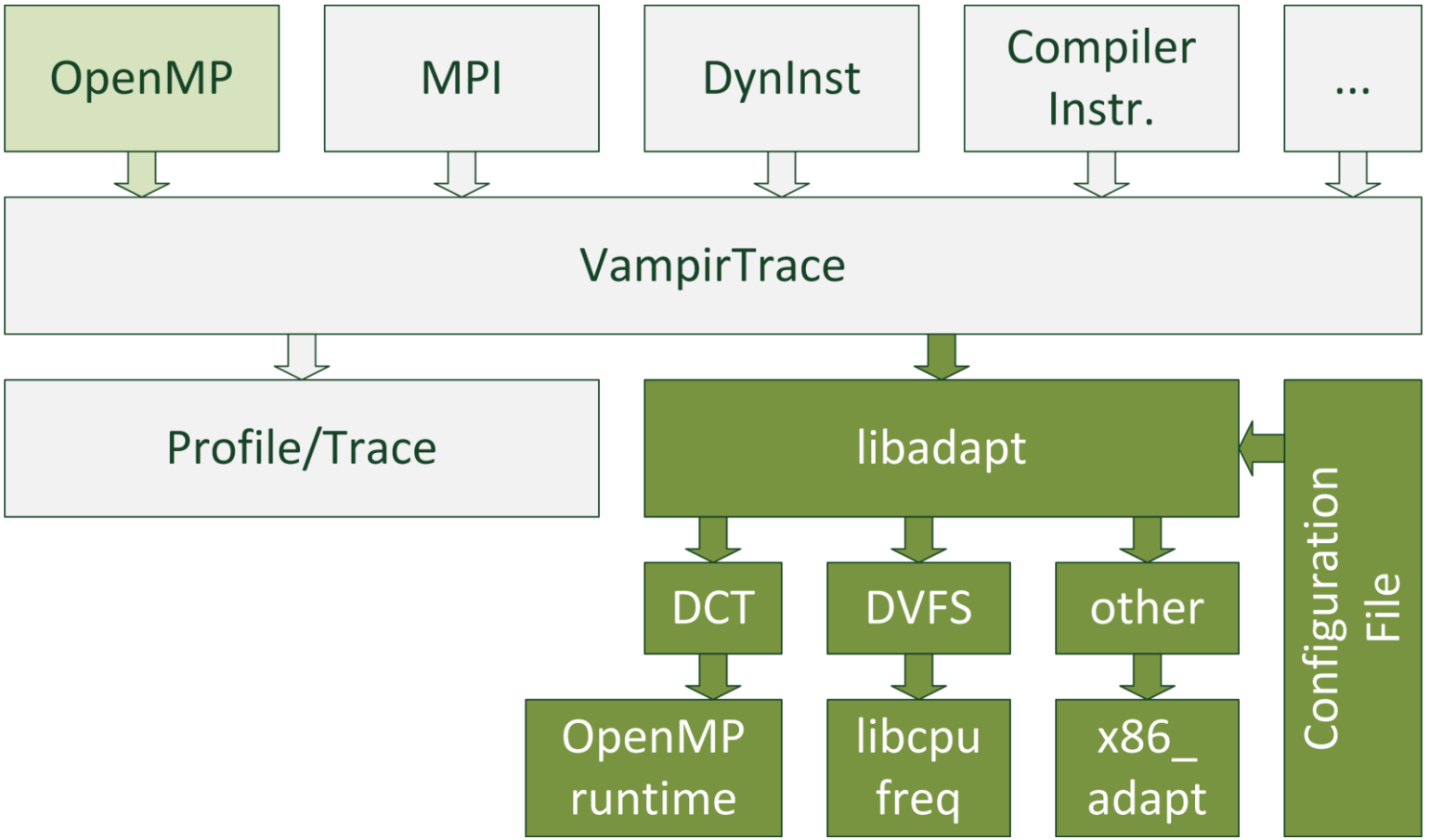
Outline

- Motivation
- Adaption Library
- Test Systems
- Evaluation
- Conclusion and Further Work

Motivation

- Power consumption of HPC systems increases
- Models for energy efficient operation points available
- How do we use these models?
  - Measure Performance Counters
  - Sampling
  - Instrumentation
- Performance analysis tools!
  - Create profiles or traces
  - Lack of the ability to change the hardware and software environment

Changes to VampirTrace



libadapt

- Uses configuration files which define adaptions
- Possible adaptions:
  - Dynamic Voltage and Frequency Scaling
  - Dynamic Concurrency Throttling
  - x86_adapt Kernel Module
- Definitions for enter and exit of functions
- Can be linked to VampirTrace

```
binary_0 :
{
name = "/home/user/my_app" ;
function_0 : {
# dct settings on the main thread
# outside of a parallel region
name = "my_function" ;
# change number of threads before
# entering  the region
dct_threads_before = 1;
# reset it afterwards
dct_threads_after = 0;
};

function_1 : {
# change non dct settings on all
# threads
name = "!$omp parallel@my_app.c:42" ;
# change frequency before and after
# entering the region
dvfs_freq_before = 1400000;
dvfs_freq_after = 2200000;
# disabling prefetcher for this region
adapt_AMD_Region_Prefetch_Disable_before
    =1;
adapt_AMD_Region_Prefetch_Disable_after
    =0;
};
};
```

Dynamic Voltage and Frequency Scaling – DVFS

- Most common energy efficiency optimization
- $P \sim V^2 * f$
- Often applied when main memory bandwidth is bottleneck
- Assumptions:
  - Reducing CPU voltage and frequency does not influence main memory performance
  - Reducing CPU voltage and frequency reduces power consumption
- Frequency decisions can be written to sysfs, operating system changes hardware configuration
- libcpufreq provides interface to sysfs files
- Problem: Main memory bandwidth is not independent of CPU frequency on all systems

Dynamic Concurrency Throttling – DCT

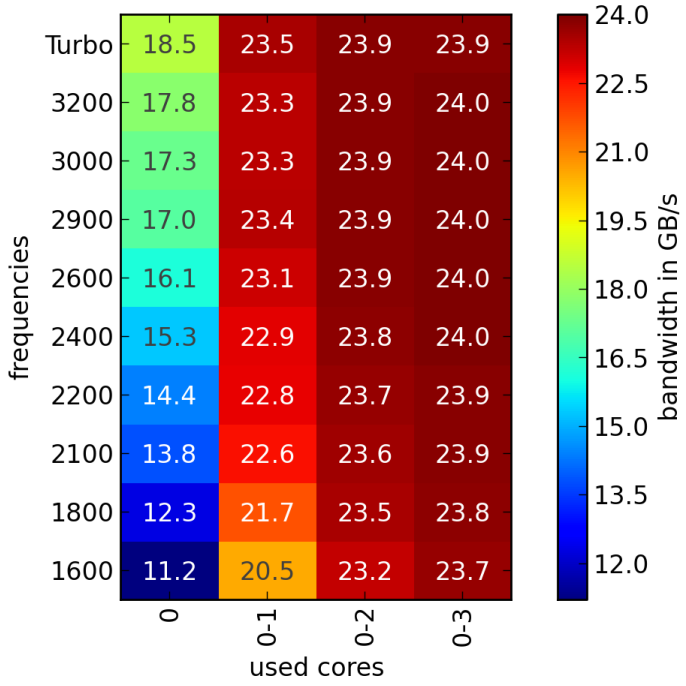
- Can be applied in thread parallel applications (OpenMP)
- If parallel regions do not scale with the number of threads, use less threads
- Software indicators: e.g. number of time spent in critical sections
- Hardware indicators: performance counters (false/true sharing, llc-misses)
- Allows processor cores to go into idle states, which allows to clock gate or power gate the cores, saving energy
- Problems:
  - Caches are flushed
  - Working sets are changed
  - Processor wakeup time can be high (depending on depth of idle state)
  - OpenMP runtime defines behavior of idling threads

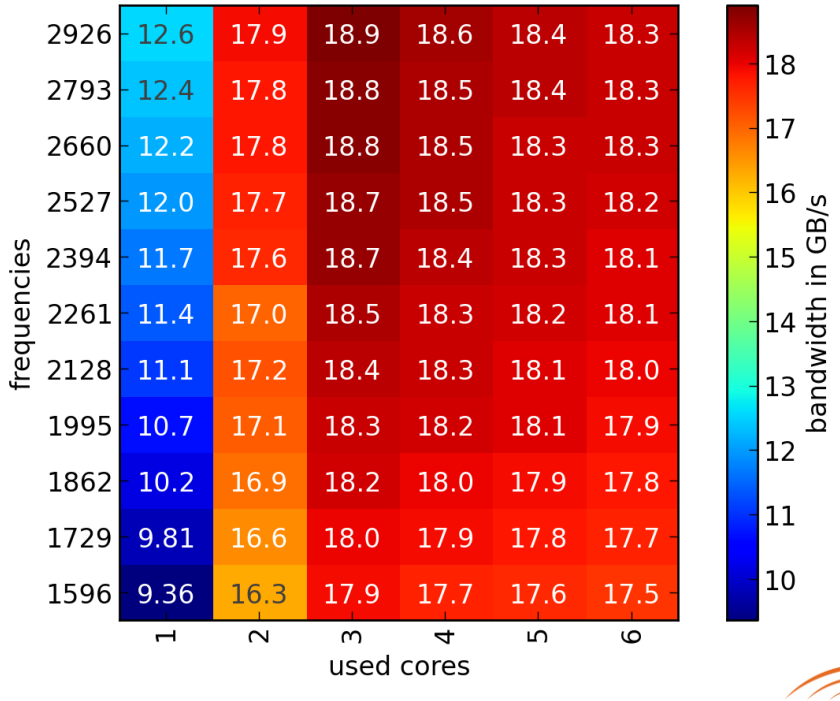
x86_adapt Kernel Module

- Provides interface for MSR and PCI based hardware configurations
- Establishes files per CPU/northbridge at /dev to change these configurations
  - allows Linux file access permissions
  - unlike msr kernel module it defines which parts of the MSRs can be accessed
- Provides additional files at /dev with the definition of available performance knobs
- E.g. prefetchers, loop predictors, …
- Known knobs compiled to module, runtime checks which ones are available for current system

Test Systems (1) – Ivy Bridge HE, Westmere EP

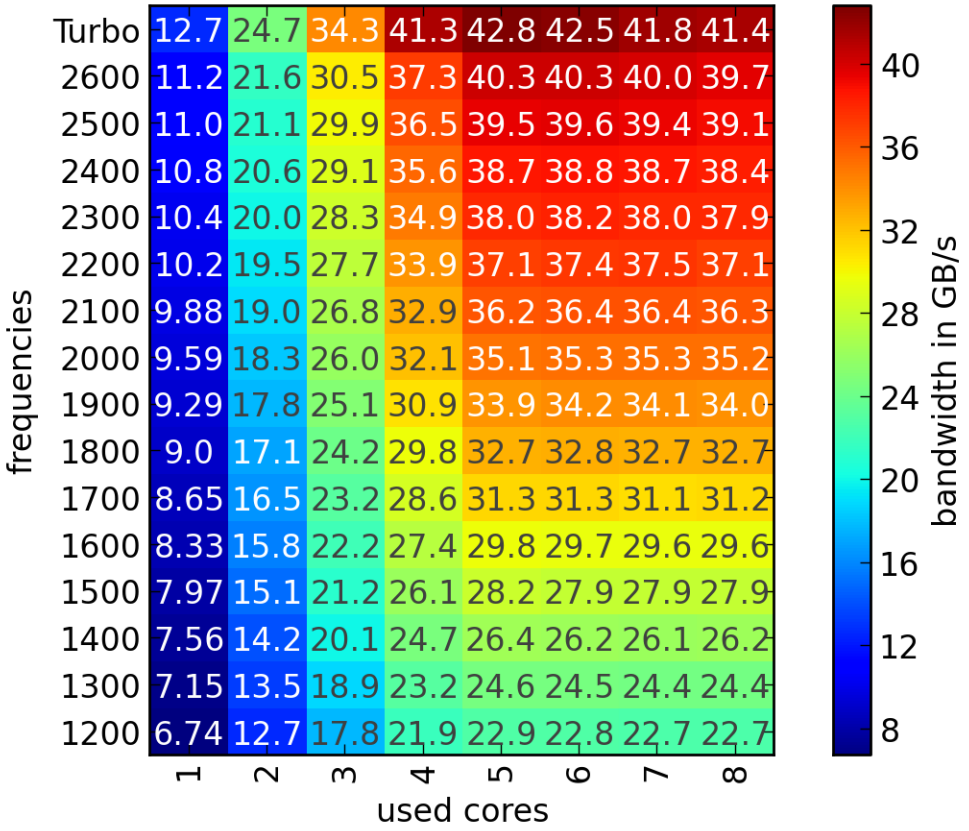
- Core i5 3470, 1 socket, 4 cores, ref. frequency: 3.2 GHz
- Xeon 5670, 2 sockets, 6 cores, ref. frequency 2.93 GHz
- Main memory bandwidth independent of CPU frequency





Test Systems (2) – Sandy Bridge EP

- Xeon E5-2670, 2 sockets, 8 cores, ref. frequency 2.6 GHz
- Main memory bandwidth depends on CPU frequency
- Max. main memory bandwidth can be achieved with less threads



Evaluation

- NAS Parallel Benchmarks (OpenMP)
  - Except dc
  - Sizes A, B, C
- Wrapper library
  - Intercepts at parallel region level
  - Provides ID of parallel regions
  - Wraps different OpenMP runtimes
- Automatic DVFS optimization
- Manual DCT optimization

Evaluation – Optimization Steps DVFS

- Run each benchmark at size B with instrumentation (wrapper+VT)
- Create performance profile, which contains runtime and performance metrics
- Based on the profile, create a configuration file with optimized DVFS settings for each parallel region
- Simple metric: Cache misses per second
- Run the benchmarks with different sizes and configurations
- Evaluate energy efficiency:

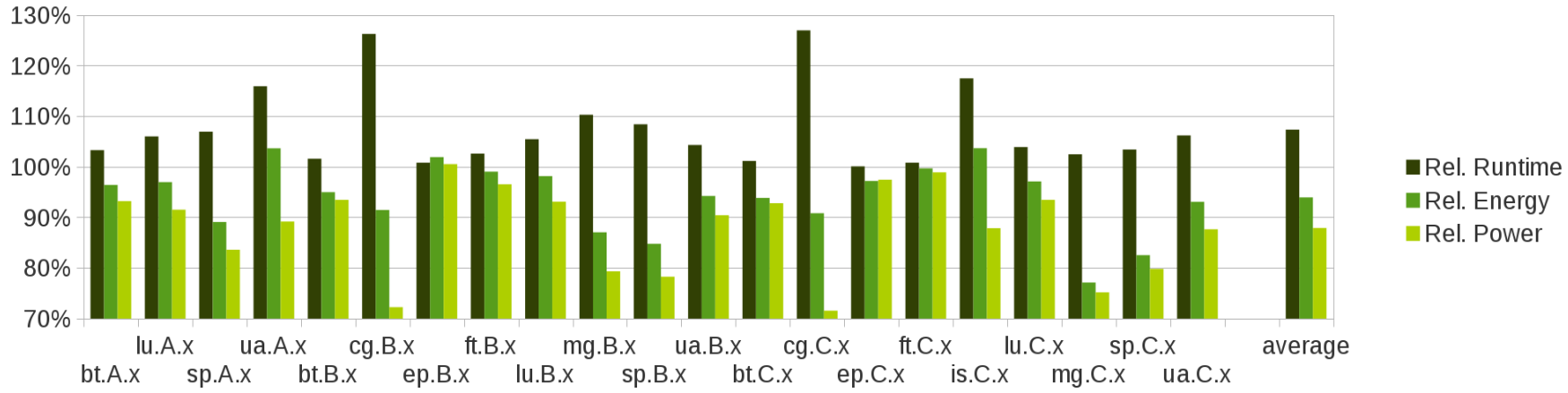
  Energy consumption of uninstrumented benchmark at reference frequency

  vs.

  instrumented benchmarks (wrapper+libadapt) with automatically generated DVFS configurations

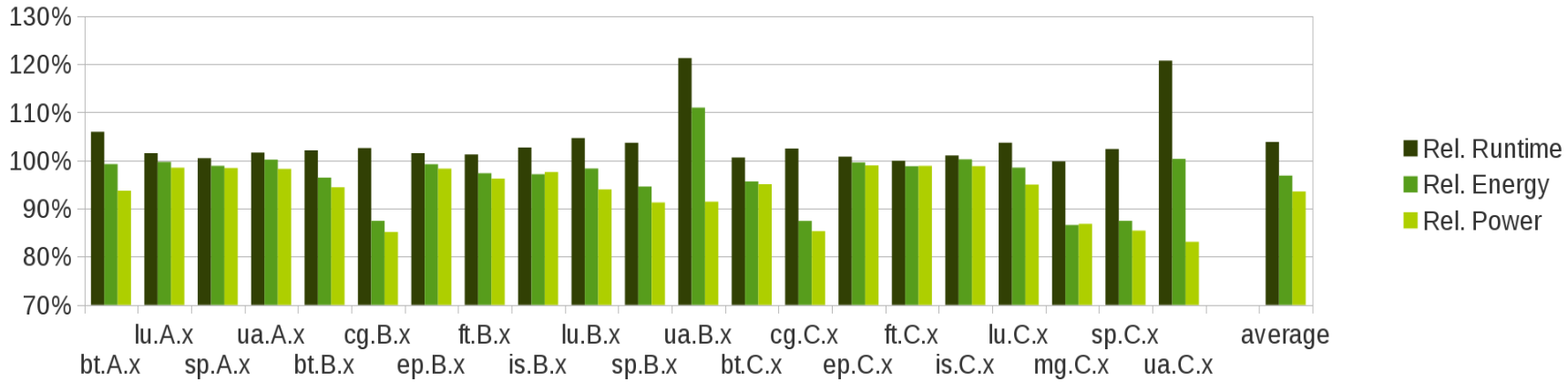
Evaluation – Ivy Bridge HE DVFS Results

- Average energy saving: 6%
- Maximal energy saving: 23%
- With increasing problem size, overhead becomes less
- Sometimes significant increase of runtime



Evaluation – Westmere EP DVFS Results

- Average energy saving: 3.2%
- Maximal energy saving: 13%
- Benchmark ua misconfigured

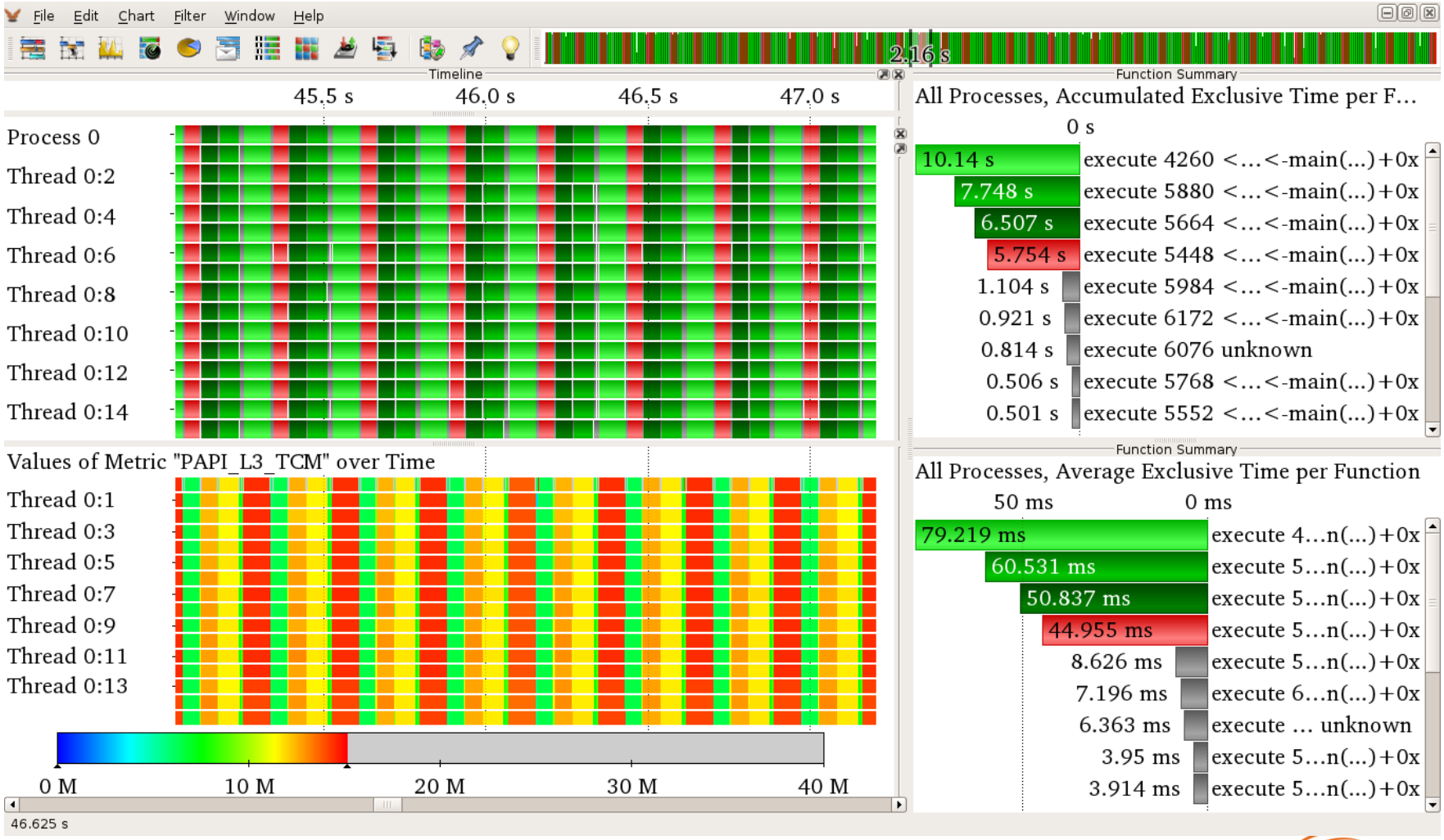


Evaluation – Optimization Steps DCT

- Manual analysis of sp.C performance trace
- Look for subsequent memory bound regions
- Reduce concurrency in these regions
- Set KMP_BLOCKTIME to 0
  - Time to busy-wait for new parallel regions
  - Set to 0 → Allows CPUs to idle
  - Default = 20 ms

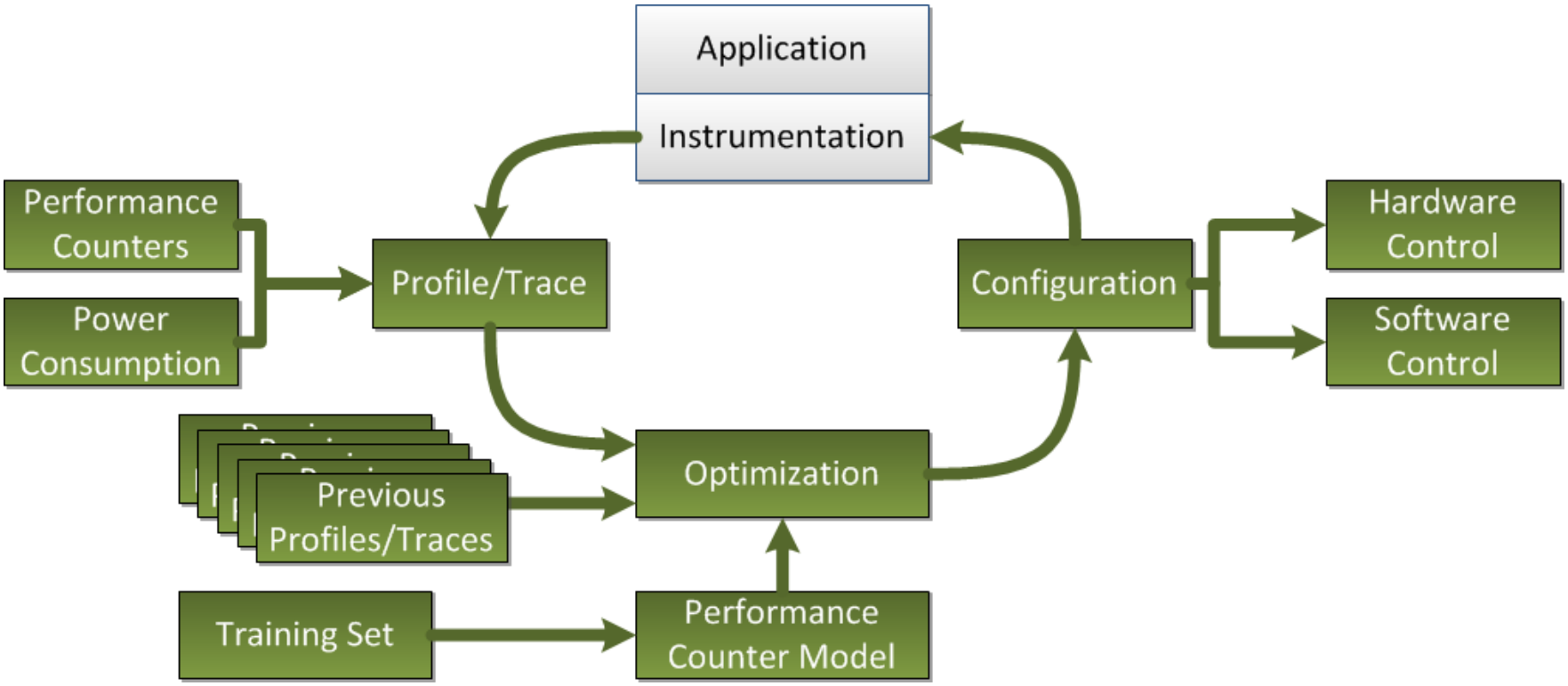
adaption	Blocktime	Runtime [s]	Energy [Ws]
None	Default	109.1	32447
	0	109.1	32094
DCT	Default	116.0 (+6.2%)	32371 (-0.2%)
	0	116.2 (+6.4%)	26534 (-18.2%)

Evaluation – Sandy Bridge DCT



Evaluation – Sandy Bridge DCT

Possible optimization loop



Conclusion and Future Work

Conclusion:

- libadapt: allows DCT, DVFS and changing of hardware settings
- Extended performance measurement tool to include libadapt
- Evaluated effectiveness with automatic DVFS optimization and manual DCT optimization
- OpenMP library wrapper allows tracing/optimizing of non-instrumented binaries

Future work

- MPI parallel optimization
- Sampling integration
- Establish optimization cycle
- Integration in Score-P

Questions?