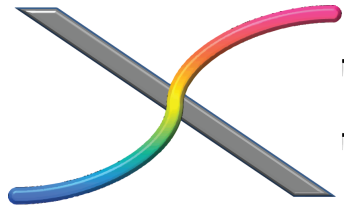


EXCESS

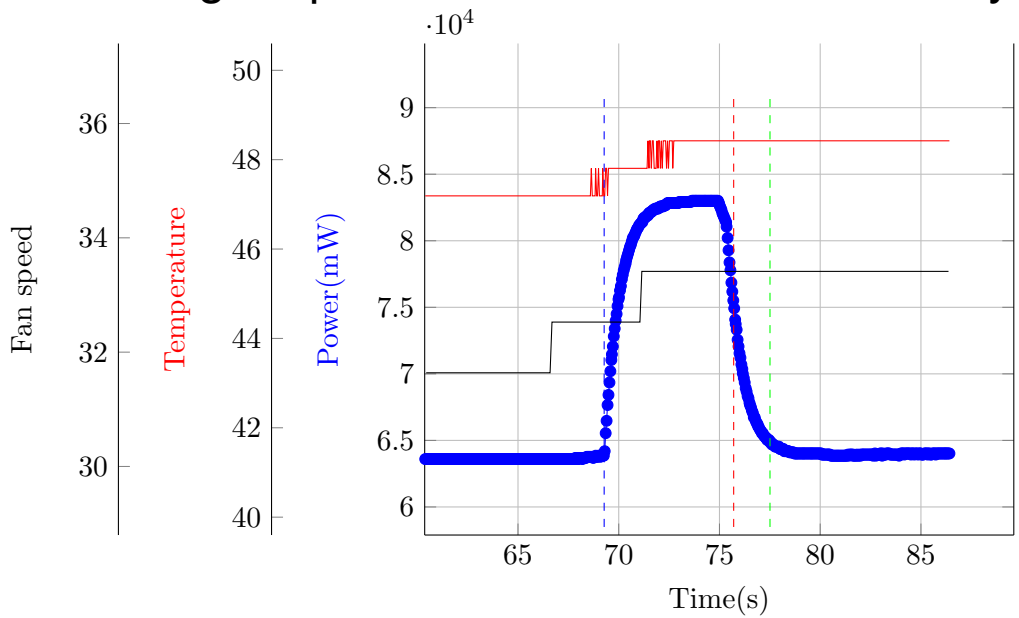
MeterPU:
A Generic Measurement Abstraction API
– Enabling Energy-tuned Skeleton Backend Selection

To appear in Proc. REPARA'15 workshop at ISPA'15, Helsinki, Finland. IEEE.

Lu Li, Christoph Kessler
lu.li@liu.se, christoph.kessler@liu.se

Motivation

- Energy Measurement
  - Necessary for energy modeling and optimization.
  - Difficult, especially for GPUs
    - Power visualization (Li et al., 2015 [4]) for a CUDA program. Illustrating "capacitor effect". Data obtained by Nvidia NVML.



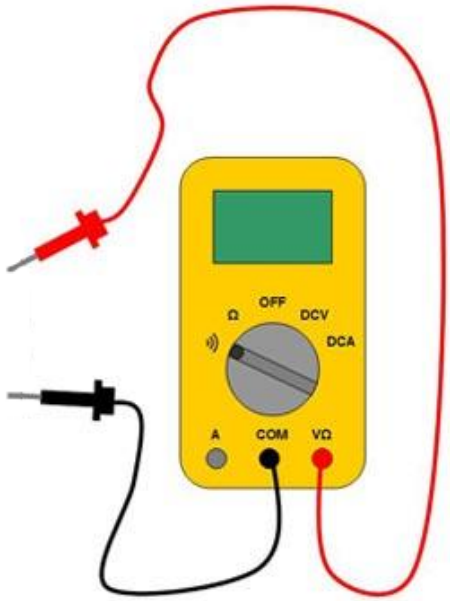
    - Requires numerical postprocessing (Burtscher et al., 2014 [1]) to calculate real energy cost
  - Goal: as simple as measuring time in good old days.

MeterPU

- A software multimeter
- A generic measurement abstraction,
  hiding platform-specific measurement details.
- Four simple functions to unify measurement interface for various metrics on different hardware components.
  - Time, Energy on CPU, GPU
  - Easy to extend: FLOPS, cache misses etc.
- On top of native measurement libraries (plug-ins).
  - CPU time: clock_gettime()
  - GPU time: cudaEventRecord();
  - CPU and DRAM energy: Intel PCM library.
  - GPU energy: Nvidia NVML library.
  - ...

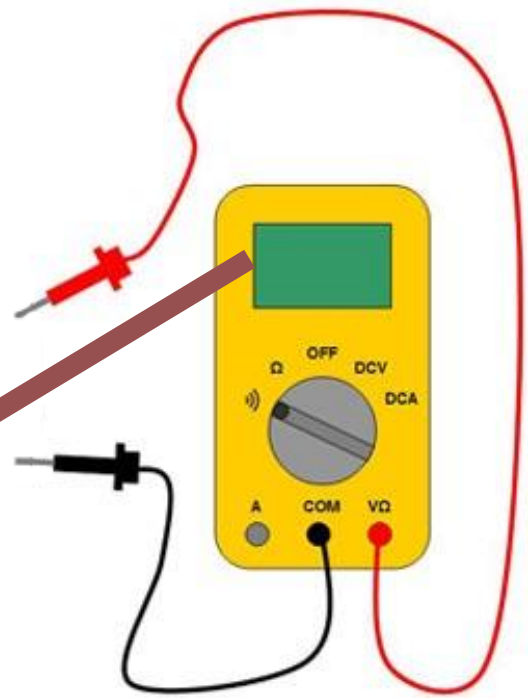
MeterPU API

```
template<class Type> //analogous to switch
                     //on a real multimeter
class Meter
{
  public:
    void start();  //start a measurement
    void stop();   //stop a measurement
    void calc();   //calculate a metric value
                   //of a code region
    typename Meter_Traits<Type>::ResultType const &
         get_value() const;
                 //get the calculated metric value
  private:
    //Platform specific measurement details hidden...
}
```



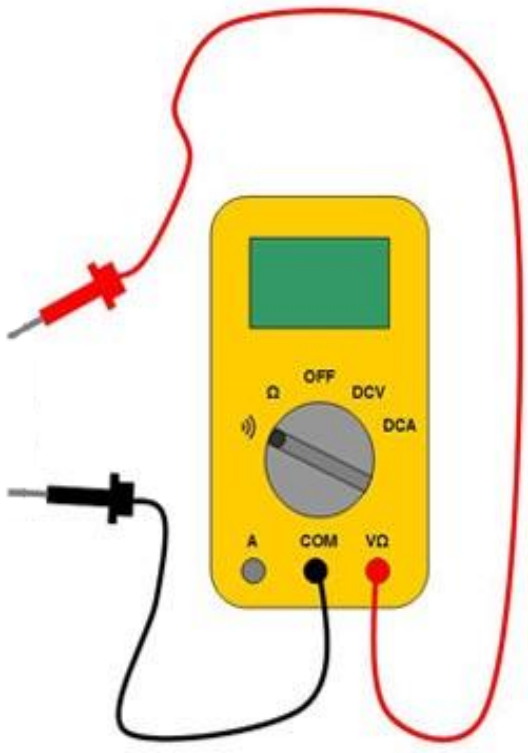
An MeterPU Application: Measure CPU Time

```
#include <MeterPU.h>

int main()
{
    using namespace MeterPU;
    Meter<CPU_Time> meter;

    meter.start();     //Measurement Start!

    cpu_func();        //Do sth here

    meter.stop();      //Measurement Stop!

    meter.calc();
    BUILD_CPU_TIME_MODEL( meter.get_value() );
}
```



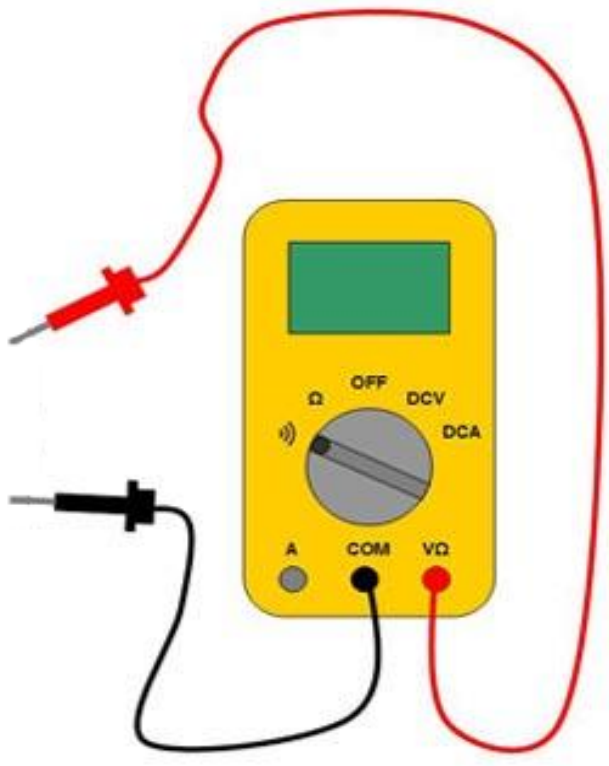
An MeterPU Application: Measure GPU Energy

```
#include <MeterPU.h>

int main()
{
    using namespace MeterPU;
    //Only one line differs!!!!
    Meter<NVML_Energy<> > meter;

    meter.start();     //Measurement Start!

    cuda_func<<< ... , ... >>>(...);
    cudaDeviceSynchronize();

    meter.stop();      //Measurement Stop!

    meter.calc();
    BUILD_GPU_ENERGY_MODEL( meter.get_value() );
}
```



Measure Combinations of CPUs and GPUs

```
#include <MeterPU.h>

int main()
{
    using namespace MeterPU;
    //Only one line differs!!!!
    Meter< System_Energy<0> > meter;

    meter.start();     //Measurement Start!

    async_cpu_func();
    cuda_func<<< ... , ... >>>(...);
    wait_for_cpu_func_to_finish();
    cudaDeviceSynchronize();

    meter.stop();      //Measurement Stop!

    meter.calc();
    BUILD_SYSTEM_ENERGY_MODEL( meter.get_value() );
}
```



Unification → Reuse Legacy Autotuning Framework

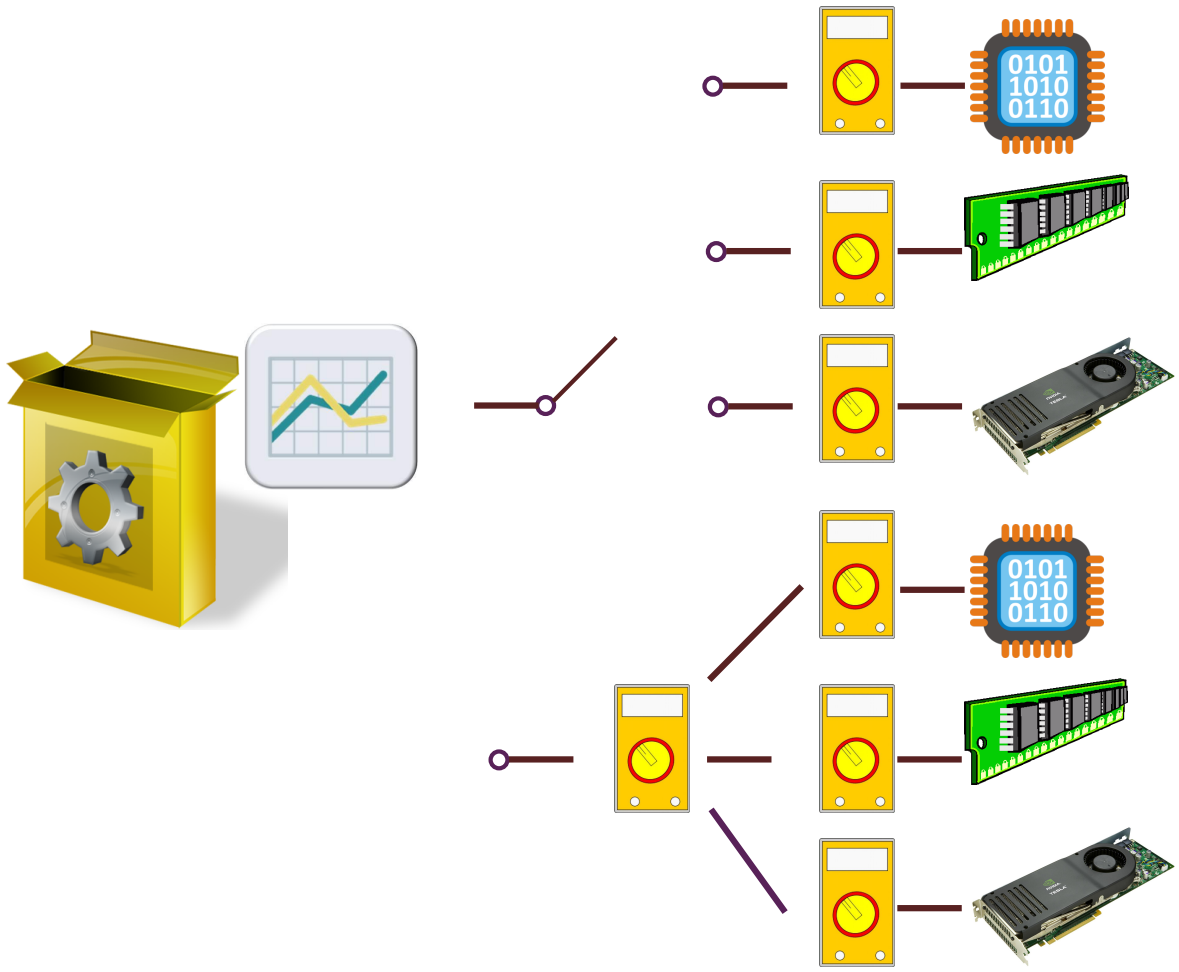
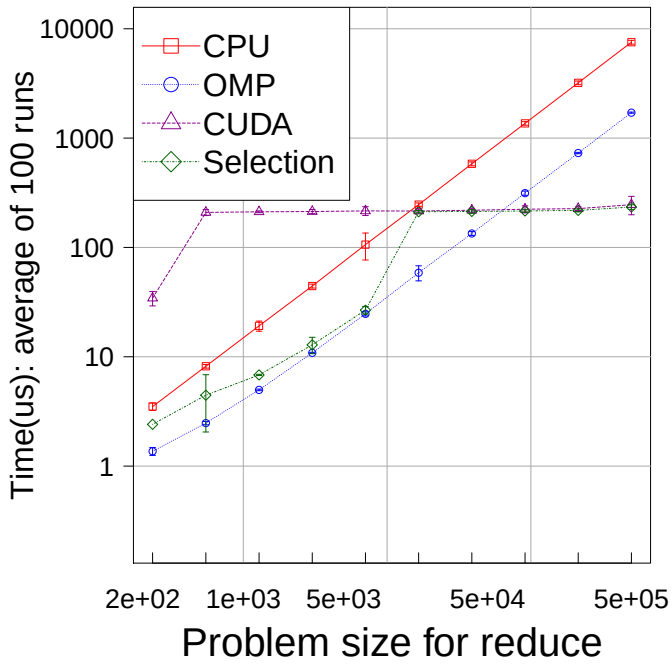
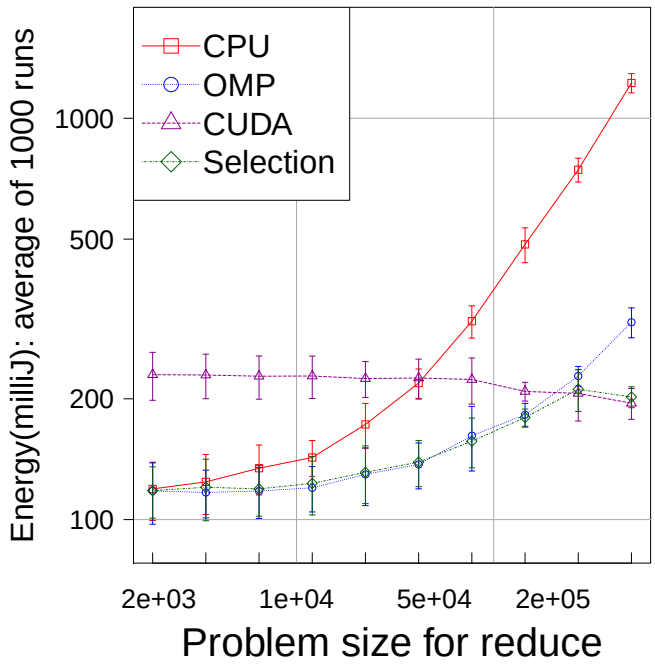


Figure 1 : Unification allows empirical autotuning framework to switch to multiple meter types on different hardware components.

SkePU Reduce Skeleton (A Single Skeleton Call)



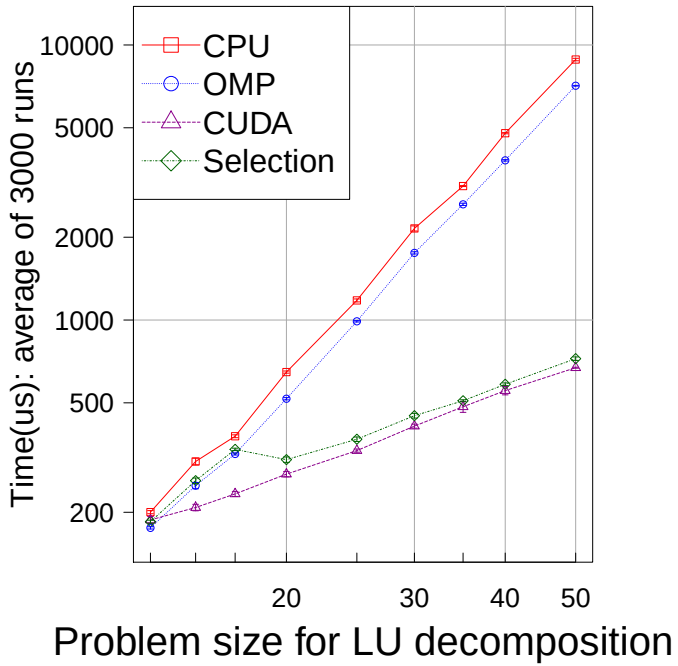
(a) Time tuning for Reduce.



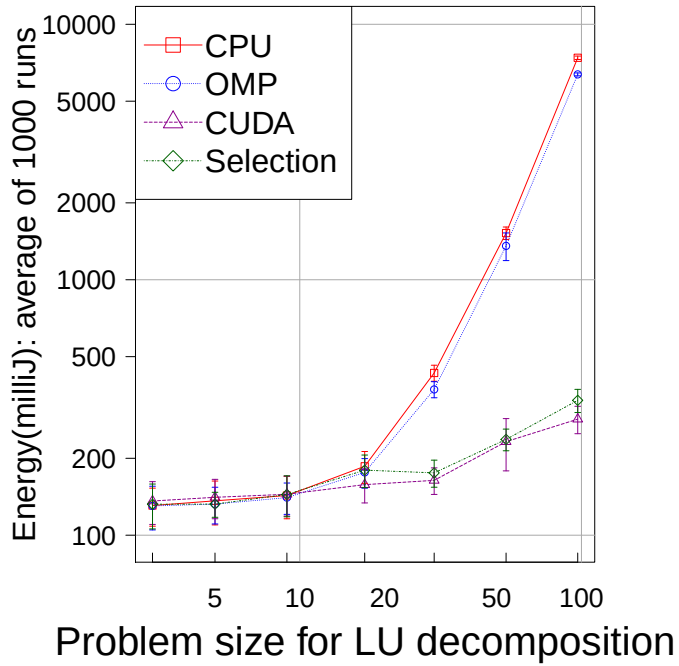
(b) Energy tuning for Reduce.

- Further results see paper
- Empirical autotuning framework for time optimization reused for energy optimization.

LU decomposition (Multiple Skeleton Calls)



(a) Time tuning for LU decomposition



(b) Energy tuning for LU decomposition

- Easy switching for optimization goals.

Related Work

- Other measurement abstraction software:
  - EML (Cabrera et al., 2015 [2])
  - REPARA (Akos et al., 2015 [3])

- MeterPU:
  - Requires much fewer lines of code (LOC) to use.
  - Very low overhead, negligible.
    - Only one function call
    - Can even be inlined for some meter types.

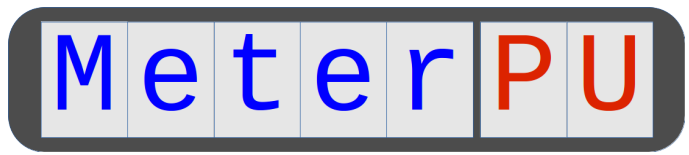
Pluggable to Other Work

- Some MeterPU Use Cases
  - Energy-tuned SkePU
  - Energy comparison between SkePU's Myriad Backend and Nvidia GPU
  - Global Composition Framework
  - TunePU
  - ...

Summary of Contributions

- Hide complexity of platform-specific energy measurement, especially for GPUs.
- MeterPU enables to reuse legacy empirical autotuning frameworks, such as the one in SkePU.
- With MeterPU, SkePU offers the first energy-tuned skeletons, as far as we know.
- Switching optimization goal can be easy, facilitates building time and energy models for multi-objective optimization.

- ☺ Open source, download at:
  `http://www.ida.liu.se/labs/pelab/meterpu`



Bibliography

Martin Burtscher, Ivan Zecena, and Ziliang Zong.

Measuring GPU power with the K20 built-in sensor.
In *Proc. Workshop on General Purpose Processing Using GPUs (GPGPU-7)*. ACM, March 2014.

Alberto Cabrera, Francisco Almeida, Javier Arteaga, and Vicente Blanco.

Energy measurement library (eml) usage and overhead analysis.
In *23rd Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP)*, pages 554–558, March 2015.

Akos Kiss, Marco Danuletto, Zoltan Herczeg, Akos Kiss, Peter Molnar, Robert Sipka, Massimo Torquati, and Laszlo Vidacs.

D6.4: REPARA performance and energy monitoring library.
Technical report, ©The REPARA Consortium, March 2015.

Lu Li and Christoph Kessler.

Validating Energy Compositionality of GPU Computations.
In *Proc. HIPEAC Workshop on Energy Efficiency with Heterogeneous Computing (EEHCO-2015)*, January 2015.