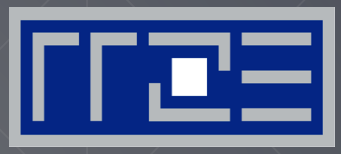
ERLANGEN REGIONAL COMPUTING CENTER



Efficient numerical simulation on multicore processors (MuCoSim) WS 2017

Prof. Gerhard Wellein, Dr. G. Hager

Department für Informatik & HPC Services
Regionales Rechenzentrum Erlangen (RRZE)

http://moodle.rrze.uni-erlangen.de/course/view.php?id=369
(see also univis)

Mission

We
care
about
performance!

- Optimization & Parallelization on all modern compute architectures

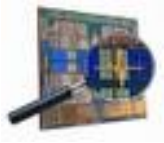
→Understand interaction between code & hardware

This is ours!

→Performance modelling: Roofline model & ECM model

→Performance tools:

likwid – lightweight performance tools
(https://github.com/RRZE-HPC/likwid)



kerncraft - Loop Kernel Analysis & Performance Modeling Toolkit
(https://github.com/RRZE-HPC/kerncraft)

- GHOST: General, Hybrid and Optimized Sparse Toolkit
(https://bitbucket.org/essex/ghost)
- ILBDC: Lattice Boltzmann Method based CFD solver

We do performance optimization, performance modeling, parallelization for

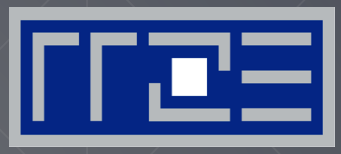
- Multi-core CPUs: core, socket, node and large scale 100,000+ cores
- GPGPUs: single devices and cluster
- Many-core CPUs: Intel Xeon Phi

- We collaborate with many users doing numerical simulation, e.g.:
  - Prof. Rüde: waLBerla / efficient C++; Prof. Fey
  - Chemistry, Physics
  - Engineering: PD Dr. S. Becker
  - Medical Image Reconstruction: Prof. Hornegger
  - Sparse Linear Solvers: Theoretical Physics, DLR; Tokyo Univ.
  - ....
- We operate the compute resources at FAU
- Our group:
  - 6 senior scientists (incl. RRZE) (GW/GH/TZ/MM/JE/MW)
  - 4 PhD students (FS/MK/TR/JH/CA)
  - 2 Master student (DE/KS)

- RRZE Testcluster („Playground“)
  - 14-core Haswell - 22-core Broadwell – AMD Desktop
  - nVIDIA & AMD GPUs & Intel Xeon Phi KNL

- RRZE production machines (Infiniband/Ominpath Interconnect):
  - **544 nodes Intel Ivy Bridge (2x 10 cores/node) → 10.880 cores**
    - **+ 8 nodes with 2 x NVIDIA K20 + 8 nodes with 2 x Intel Xeon Phi / KNC**
  - **728 nodes Intel Broadwell (2x10 cores / node) → 14.560 cores**

- Access to external machines
  - NIC Jülich: IBM BlueGene/Q 450,000+ cores (6 PFLOP/s)
  - LRZ Garching: Intel Cluster ((3.2+3.6) PFLOP/s; 230,000 cores)
  - HLR Stuttgart: CRAY XC40 (7.4 PFLOP/s; 185,000 cores)
  - CSCS Lugano: CRAY XC50 (4,500 nodes with 1 CPU+1NVIDIA P100)
  - Oakforrest-PACS (Japan): Intel KNL (8,000 nodes with KNL only)



MuCoSim SS 2017

Format

„Old“ talks from previous semester;

New projects:

- „Journal Club“
- Focus on stencil optimization and modelling

Projects from WS 2016/7 – 2nd talks from last semester

- T. Köster – „Roofline and ist roots“
- D. Uhl – „Energy-Roofline-Model“
- A.Hariharan – „Checksum-based silent error detection in CG-Algorithm"

Stencil schemes

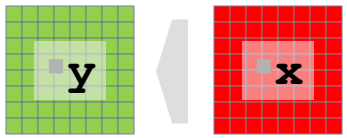
- Stencil schemes frequently occur in PDE solvers on regular lattice structures
- Basically it is a sparse matrix vector multiply (spMVM) embedded in an iterative scheme (outer loop)
- but the regular access structure allows for matrix free coding

```
do iter = 1, max_iterations

  Perform sweep over regular grid: y(:) ← x(:)

  Swap y ←→ x

enddo
```



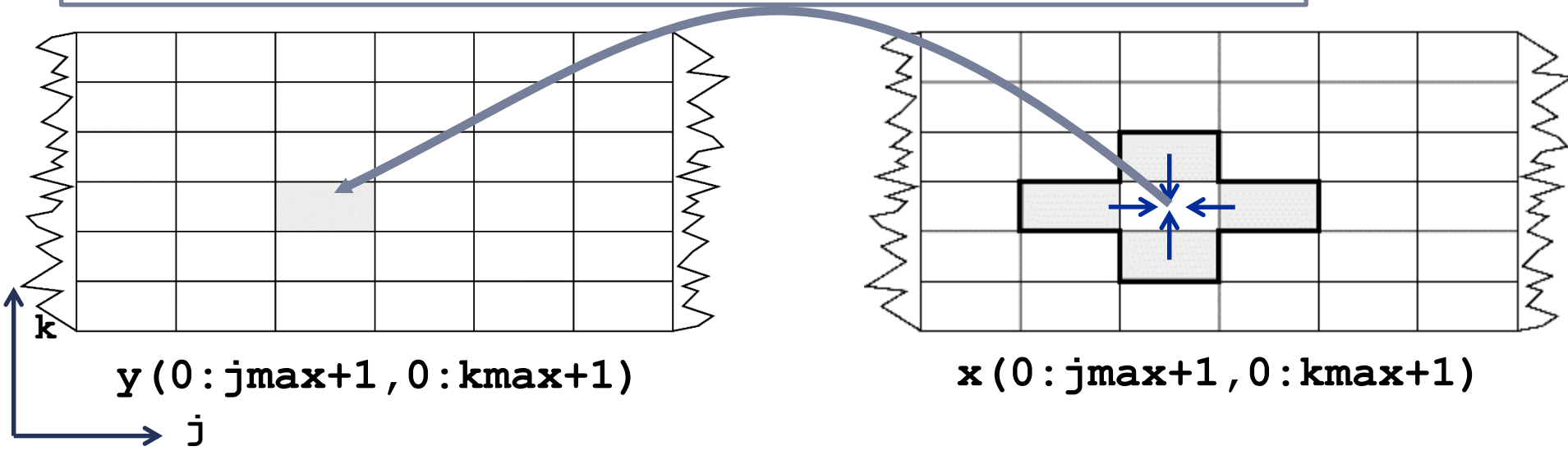
- Complexity of implementation and performance depends on
  - stencil operator, e.g. Jacobi-type, Gauss-Seidel-type, …
  - spatial extent, e.g. 7-pt or 25-pt in 3D,…

Jacobi-type 5-pt stencil in 2D

sweep

```
do k=1,kmax
  do j=1,jmax
    y(j,k) = const * (  x(j-1,k) + x(j+1,k) &
                      +    x(j,k-1) + x(j,k+1) )
  enddo
enddo
```

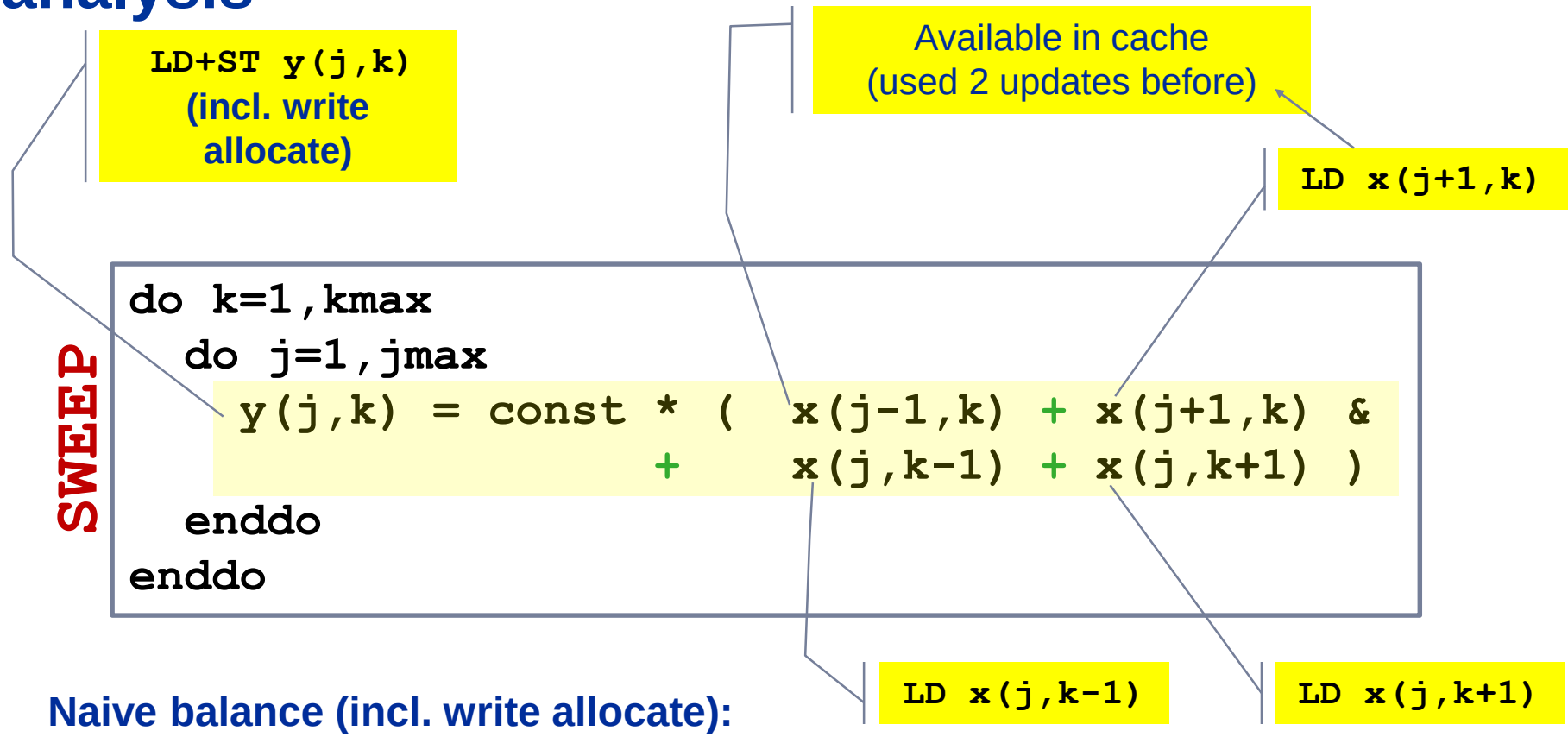
- Lattice Update (LUP)



Appropriate performance metric: "**Lattice Updates per second**" [**LUP/s**]

(here: Multiply by 4 FLOP/LUP to get FLOP/s rate)

Jacobi 5-pt stencil in 2D: data transfer analysis



Naive balance (incl. write allocate):

`x( :, :)` : 3 LD +
`y( :, :)` : 1 ST+ 1LD

→ B_C = 5 Words / LUP = 40 B / LUP (assuming double precision)

What to do for stencils

Set up (analytical) model:

1. How much data travels between different memory levels
2. Make Roofline estimate
3. Establish ECM model – use brain & kerncraft

Validate model predictions by

1. Performance measurements
2. Data traffic measurements → likwid-perfctr

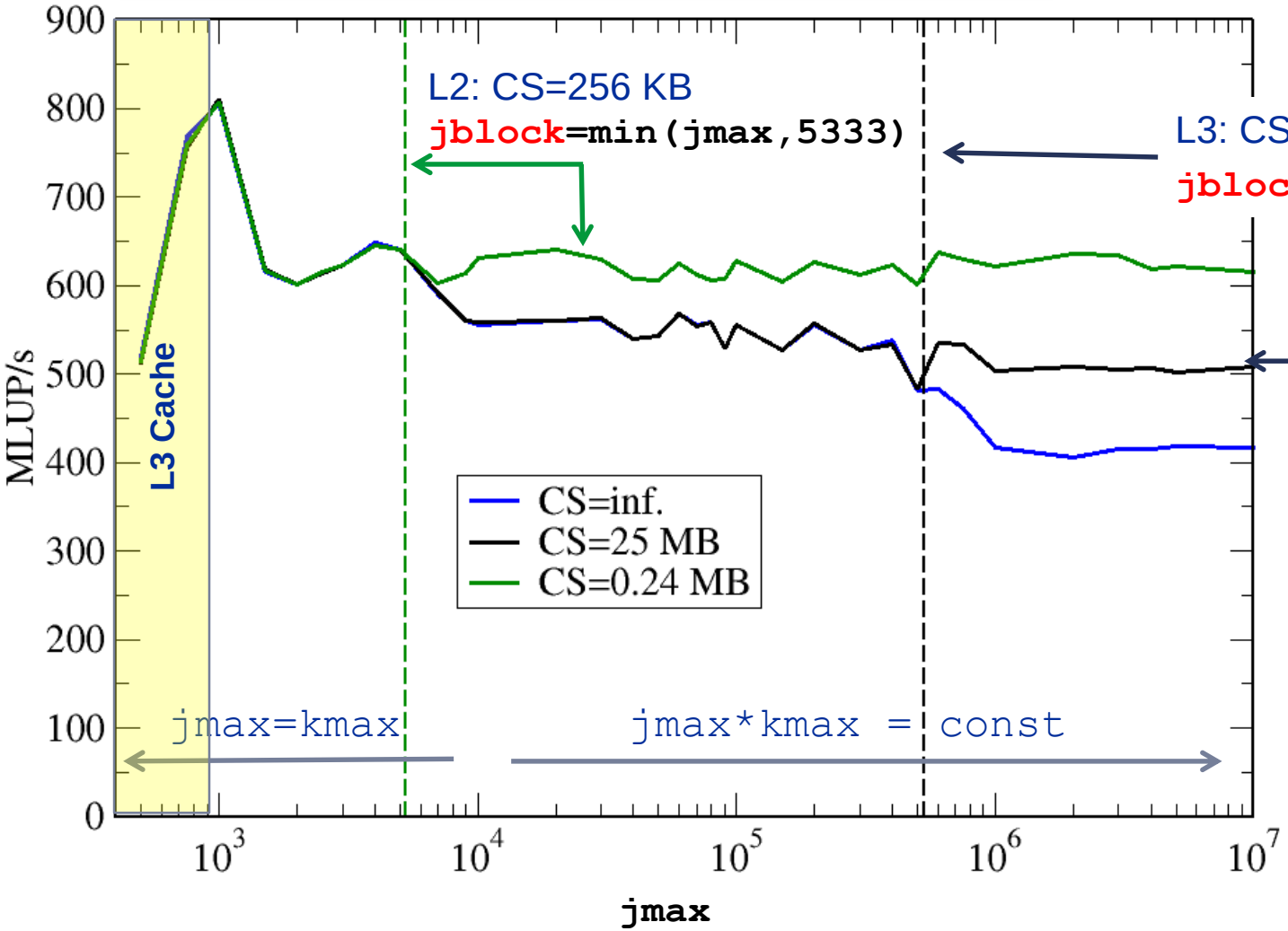
Choose appropriate code optimization technique, e.g.

- Spatial blocking
- Temporal blocking
- Semi Stencil & Related

Establish layer condition by spatial blocking

`jblock < CacheSize / 48 B`

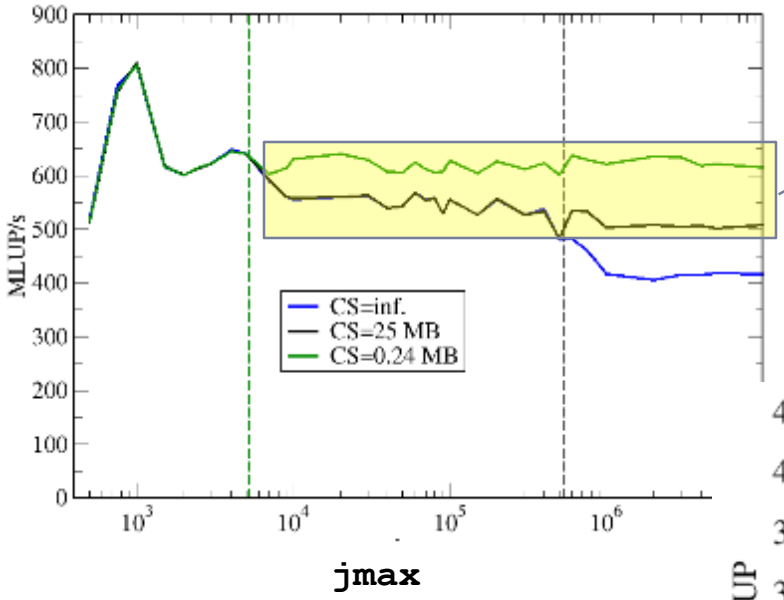
Which cache to block for?



Intel Compiler
ifort V13.1
Intel Xeon E5-2690 v2
("IvyBridge"@3 GHz)

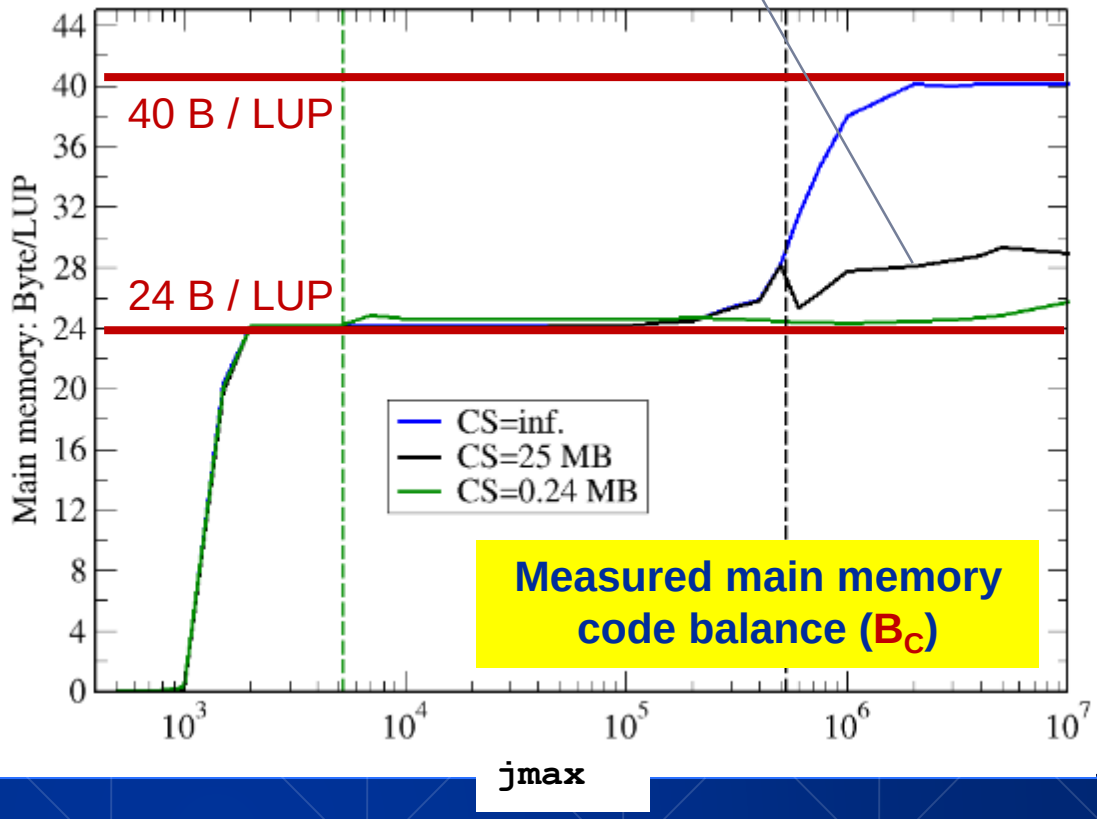
L1: 32 KB
L2: 256 KB
L3: 25 MB

Validating the model: Memory code balance



Main memory access is not reason for different performance (but L3 access is!)

Blocking factor (CS=25 MB) still a little too large

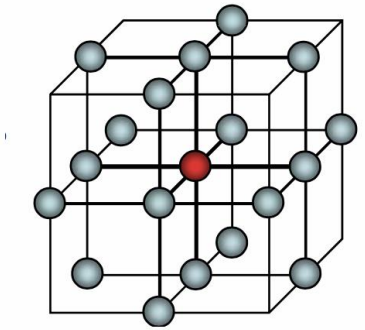


Measured main memory code balance (B_C)

Intel Compiler
ifort V13.1
Intel Xeon E5-2690 v2
("IvyBridge"@3 GHz)

Stencil topics – Himeno benchmark (2 students)

The Riken Himeno CFD Benchmark:
http://accc.riken.jp/HPC/HimenoBMT/index e.html



```
for (i=1; i<imax-1; i++)
   for (j=1; j<jmax-1; j++)
      for (k=1; k<kmax-1; k++)
{
 s0 =  a0[i][j][k]* p[i+1][j][k]
     + a1[i][j][k]* p[i][j+1][k]
     + a2[i][j][k]* p[i][j][k+1]
     + b0[i][j][k]*(p[i+1][j+1][k] - p[i+1][j-1][k]
                  - p[i-1][j+1][k] + p[i-1][j-1][k])
     + b1[i][j][k]*(p[i][j+1][k+1] - p[i][j+1][k-1]
                  - p[i][j-1][k+1] + p[i][j-1][k-1])
     + b2[i][j][k]*(p[i+1][j][k+1] - p[i+1][j][k-1]
                  - p[i-1][j][k+1] + p[i-1][j][k-1])
     + c0[i][j][k]* p[i-1][j][k]
     + c1[i][j][k]* p[i][j-1][k]
     + c2[i][j][k]* p[i][j][k-1]
     + wrk1[i][j][k];
 ss = (s0 * a3[i][j][k]-p[i][j][k])
     * bnd[i][j][k];                    //(ss = delta P)
 wrk2[i][j][k]=p[i][j][k]+omega*ss;     //(over-relaxation)
 gosa += ss*ss;                         //(residual, measure of convergence)
}
```

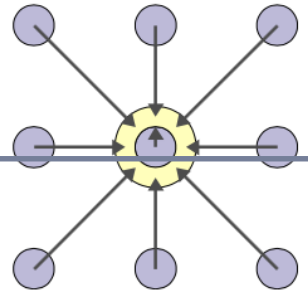
Implementation
ECM Model
Validation
Temporal Blocking → Girih*
Target: Xeon & Xeon Phi/KNL

Code&figure: Phillips & Fatica, Implementing the Himeno benchmark with CUDA on GPU clusters, IEEE IPDPS 2010

*T. Malas, G. Hager, H. Ltaief, H. Stengel, G. Wellein, and D. Keyes, **SIAM SISC 2015**

Advisor: GH / GW

Stencil topics – Box stencils & Semi-stencil



```
do k=1,kmax
  do j=1,jmax
    y(j,k) = x(j,k)+c10*x(j-1,k) +  c20*x(j+1,k) &
                   +c01*x(j,k-1) +  c02*x(j,k+1) &
                   +c11*x(j-1,k-1)+ c21*x(j+1,k-1)&
                   +c21*x(j-1,k+1)+ c22*x(j+1,k+1)
  enddo
enddo
```

Tasks:

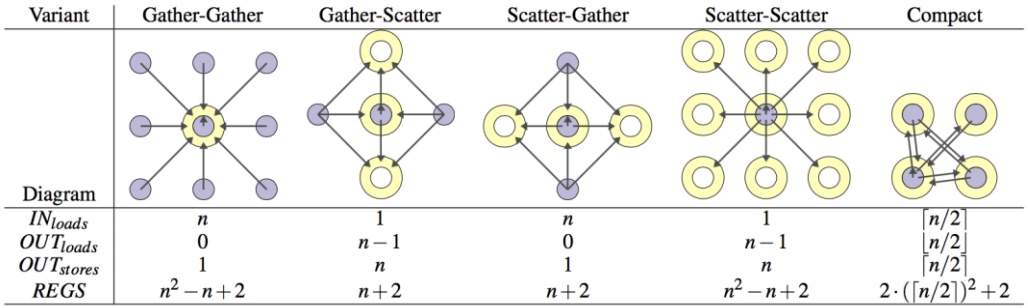
- Implement OpenMP paralölel box stencil in 2D & 3D
- Roofline/ECM; modelling – spatial blocking?!
- Apply semi stencil optimization

Advisor: GW / JH

Stencil automatic compiler optimizations

A Framework for Enhancing Data Reuse via Associative Reordering

Kevin Stock[1] Martin Kong[1] Tobias Grosser[2]
Louis-Noël Pouchet[3] Fabrice Rastello[4] J. Ramanujam[5] P. Sadayappan[1]



Variant	Gather-Gather	Gather-Scatter	Scatter-Gather	Scatter-Scatter	Compact
Diagram					
IN_{loads}	n	1	n	1	$\lceil n/2 \rceil$
OUT_{loads}	0	$n-1$	0	$n-1$	$\lfloor n/2 \rfloor$
OUT_{stores}	1	n	1	n	$\lceil n/2 \rceil$
$REGS$	n^2-n+2	$n+2$	$n+2$	n^2-n+2	$2\cdot(\lceil n/2 \rceil)^2+2$

- Tasks:
  Install framework
- Reproduce performance results
- Roofline/ECM Model for selected code transformations.

See Stock et al, PLDI '14, June 9-11 2014, Edinburgh, UK.
http://dx.doi.org/10.1145/2594291.2594342

Advisor: GH / CA

ECM & performance of standard stencils on Haswell/Broadwell (2 students)

Stengel et al., *Quantifying performance bottlenecks of stencil computations using the Execution-Cache-Memory model*. Proc. ICS15. DOI: 10.1145/2751205.2751240

Malas et al., *Multicore-optimized wavefront diamond blocking for optimizing stencil updates*. SIAM SISC 2015. DOI: 10.1137/140991133

Tasks:

- Implement a OpenMP parallel benchmark framework which tests all the stencils in the 2 papers
- Comprehensive benchmarking – architecture, scalability, problem size (Haswell/Broadwell)
- ECM/Roofline model – Validation using HW performance counter
- Spatial blocking/ Modelling & Validation

Advisor: GH / GW

Performance of standard stencils on NVIDIA GPU

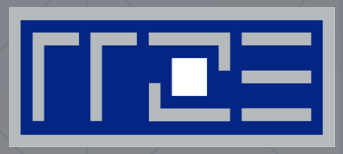
Stengel et al., *Quantifying performance bottlenecks of stencil computations using the Execution-Cache-Memory model*. Proc. ICS15. DOI: 10.1145/2751205.2751240

Malas et al., *Multicore-optimized wavefront diamond blocking for optimizing stencil updates*. SIAM SISC 2015. DOI: 10.1137/140991133

Tasks:

- Implement selected stencils on NVIDIA GPU
- Comprehensive benchmarking
- Check layer conditions using HW performance counter
- Spatial blocking

Advisor: MK / DE



Non-Stencil Topics

Integrating CRAFT functionality in CARP-CG algorithm in PHIST and benchmarking on SuperMUC/Piz-Daint

CRAFT:

1. Application-level Checkpoint/Restart(CR):
   - A simple and **extendable** interface for enabling applications to integrate CR functionality with minimal code changes.
   - Most frequently used data-types can be used out of the box; e.g., plain-old-data(POD), POD Arrays & multi-arrays.

2. Automatic Fault Tolerance (AFT):
   - Defines `AFT-zones´ for automatic communication recovery in case of process failure(s).
   - Recovery models: shrinking, non-shrinking (spawn+merge)

```
#include <craft.h>
int main(int argc, char* argv[])
{
  ...
  int myrank;
  MPI_Comm FT_Comm;
  MPI_Comm_dup(MPI_COMM_WORLD, &FT_Comm);
  AFT_BEGIN(FT_Comm, &myrank, argv);

  double data = 0;
  int iteration = 0, cpFreq = 10;
  Checkpoint myCP( "myCP", FT_Comm);
  myCP.add("data", &data);
  myCP.add("iteration", &iteration);
  myCP.commit();

  if( myCP.needRestart() ) {myCP.read();}
  for(; iteration <= n; iteration++)
  {
    // Computation-communication loop
    if(iteration % cpFreq == 0)
      { myCP.update(); myCP.write();}
  }
  ...
  AFT_END();
}
```

Advisor: FS

Intel Pin and SDE

- Pin: Dynamic Binary Instrumentation Tool
  - Intercept assembly and insert code
    - Count specific instruction/function calls
    - Transform one instruction to another (AVX load to split SSE loads)
    - https://software.intel.com/en-us/articles/pintool/
- SDE: Software Development Emulator
  - Run code in a virtual architecture
  - Sets up on top of Pin
  - https://software.intel.com/en-us/articles/intel-software-development-emulator/
- Questions for the seminar: How much work is it to use Pin? Where are the limitations ? What's the overhead of running an application with Pin and SDE? How is the accuracy?

Advisor: TR

Journal club - What to do?

Goal: A critical review of important papers in the field of High Performance Computing

1. Read the paper(s) assigned to you thoroughly
2. Discuss open questions with your advisor.
3. In collaboration with your advisor, perform programming and/or measurements of your own (depends on the topic)
4. Prepare slides with a summary of your paper(s) and experiments (one or two talks <30min each)
   - Concentrate on main insight, not on every little bit of data
   - If the paper contains BS, say so (and provide proof!)
5. Every seminar attendee should be able to understand the main findings of the paper(s) you present

Stencil optimizations

Stencil algorithms appear frequently in scientific computing. Optimization of those algorithms is the subject of intense research.

- Frigo et al.: Cache Oblivious Stencil Computations. Proceedings of the 19th annual international conference on Supercomputing, Pages 361-366. DOI: 10.1145/1088149.1088197
- D. Wonnacott: Achieving Scalable Locality with Time Skewing. International Journal of Parallel Programming, Vol. 30, No. 3, June 2002. DOI: 10.1023/A:1015460304860

Stencil DSLs (2 students)

Domain-specific languages (DSLs) are regarded by many as a solution to the optimization problem with stencil algorithms.

- Y. Tang et al.: The Pochoir Stencil Compiler. SPAA '11 Proceedings of the twenty-third annual ACM symposium on Parallelism in algorithms and architectures, p. 117-128. DOI: 10.1145/1989493.1989508
- Y. Tang et al.: Coding Stencil Computations Using the Pochoir Stencil-Specification Language. USENIX HotPar'11 May 26–27, 2011, Berkeley, California, USA
- U. Bondhugula et al.: Automatic Transformations for Communication-Minimized Parallelization and Locality Optimization in the Polyhedral Model International Conference on Compiler Construction (ETAPS CC), Apr 2008, Budapest, Hungary.
- U. Bondhugula et al.: A Practical Automatic Polyhedral Parallelizer and Locality Optimizer. ACM SIGPLAN Programming Languages Design and Implementation (PLDI), Jun 2008, Tucson, Arizona.

Modeling by fitting and machine learning

Analytic modeling can be complemented by machine learning approaches that automatically predict metrics (runtime, energy consumption) based on code features or hardware counter measurements.

- A. Tiwari et al.: Modeling Power and Energy Usage of HPC Kernels. Proc. IPDPS 2012 Workshops, DOI: 10.1109/IPDPSW.2012.121
- J. Peraza et al.: Understanding the Performance of Stencil Computations on Intel's Xeon Phi. Proc. Cluster 2013, DOI: 10.1109/CLUSTER.2013.6702651
- A. Calotoiu et al.: Using Automated Performance Modeling to Find Scalability Bugs in Complex Codes. Proc. SC13, DOI: 10.1145/2503210.2503277

Benchmarking & best practices

This paper proposes twelve best practices for measuring and presenting performance data. It can be seen as a complement to the “Fooling the Masses” motif.

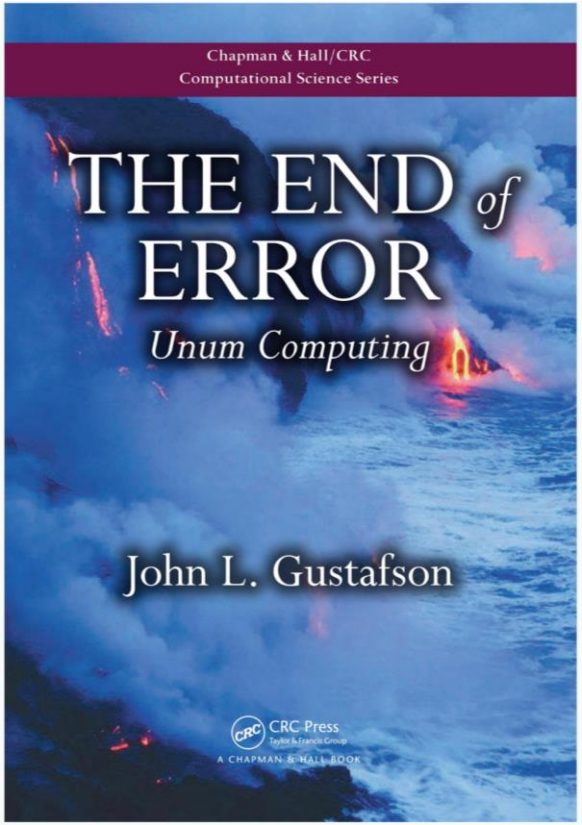
- T. Hoefler et al.: Scientific Benchmarking of Parallel Computing Systems. Proc. SC15, DOI: 10.1145/2807591.2807644

The “Unum” number format

“Unum” was proposed as a solution to the ubiquitous accuracy problem with floating-point representations.

- J. L. Gustafson: The End of Error: Unum Computing.
- J. L. Gustafson: A radical approach to computation with real numbers. DOI: 10.14529/jsfi160203

- See also http://johngustafson.net/unums.html



Structure of the seminar

- Introduction to RLM/ECM (GW/GH)
- Basic stencil modelling (GW/GH)
- Introduction to likwid (TR)
- Introduction to kerncraft (JH)

- Talks from last semester

- Talks from master students (KS, DE)

- Potentially talks by external experts

Tools to use from our group

General

- likwid-perfctr → Thomas Röhl
- kerncraft → Julian Hammer

Sparse solvers:

- CRAFT / PHIST → Faisal Shazad
- GHOST / PHIST → Moritz Kreutzer