Technische Universität München
Institut für Informatik
Prof. Dr. Hans-Joachim Bungartz
Alexander Heinecke, M.Sc., M.Sc.w.H.

WS 2010/2011

Algorithms of Scientific Computing II

Exercise 4 - Hardware-aware implementations and Barnes-Hut-Method

1) Today's Systems - Compute Power

Today's supercomputer are highly parallel systems. Please read LRZ's web page and identify the different levels of parallelism in SuperMUC!

ANSWER:

Vector-register; Hardware-thread; core; processor; node; island; whole system.

Which levels of parallelism are not documented there, but can be found in Intel Architecture Manual?

ANSWER:

Instruction level parallelism: A core is able to decode several instruction a time (e.g. four-way decode). Since it features several execution units the so-called out-of-order unit is able to run different instructions employing different units out-of-order and in parallel. Please refer to Fig. 1 and Fig. 2 for examples.

In total we are now able to calculated the peak-performance (double percision) of a node in SuperMUC.

- **2**: out-of-order: mul and add
- **4**: vectorization
- **8**: cores per package
- **2.7**: clock-speed
- **2**: two sockets per node

$\Rightarrow$ 345.6 GFLOPS per node! $\Rightarrow 345.6 \cdot 512 \cdot 18 = 3.19$ PFLOPS for whole system.

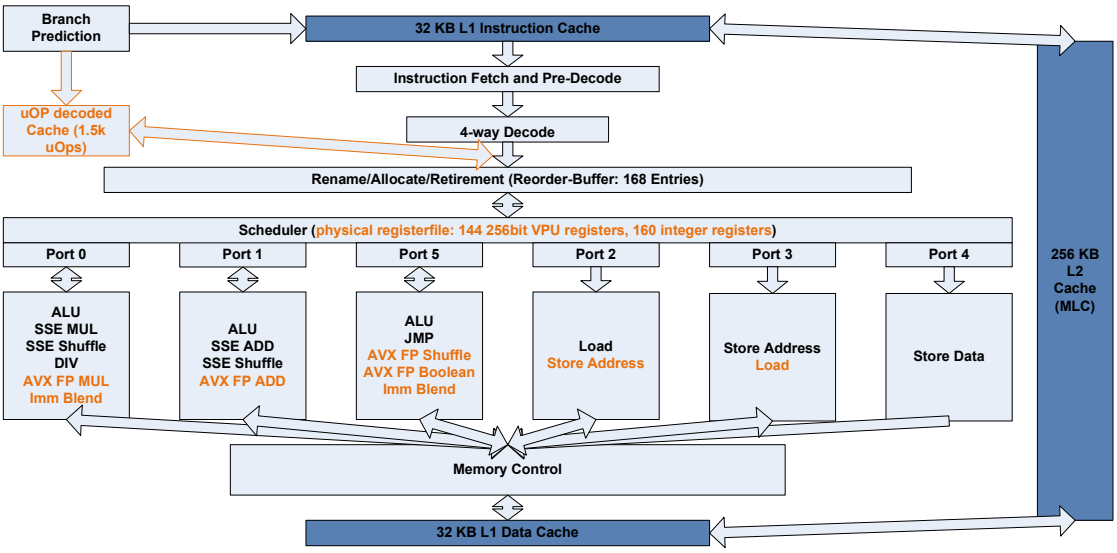

Abbildung 1: Intel Sandy Bridge Architecture, changes in comparison to Intel Nehalem Architecture are highlighted in orange, based on descriptions from Intel.

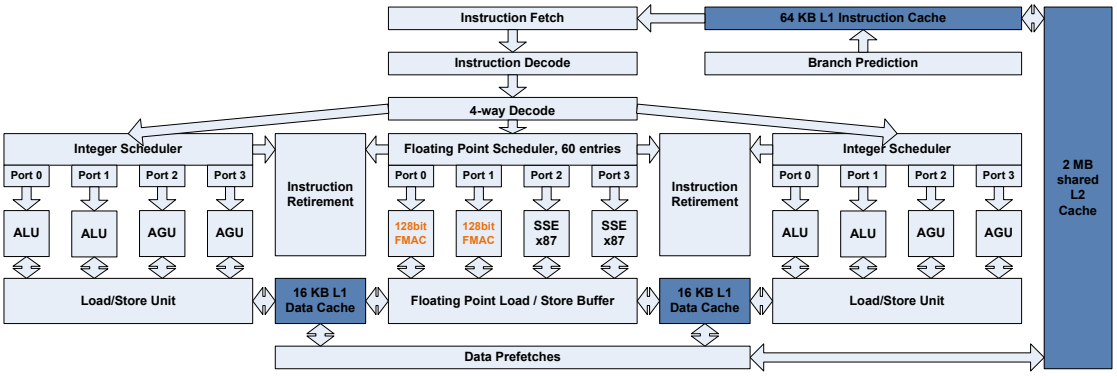

Abbildung 2: AMD Bulldozer Architecture, based on descriptions from AMD.

2) Today's Systems - Memory Subsystem

Please sketch the memory subsystem/ network structure of SuperMUC. Is there a different network topology available on another supercomputer?

ANSWER:

A single node features a state of the art NUMA-architecture as displayed in Fig. 3, which implements a shared memory but with different access time: local vs. remo-

te. A remote access takes roughly 10-20% longer than a local access. Operating systems like Linux employ a so called first touch policy: the NUMA-node which touches a page first is the owner of the this page. On each socket we are able to achieve a streaming bandwidth of more than 40 GByte/s.

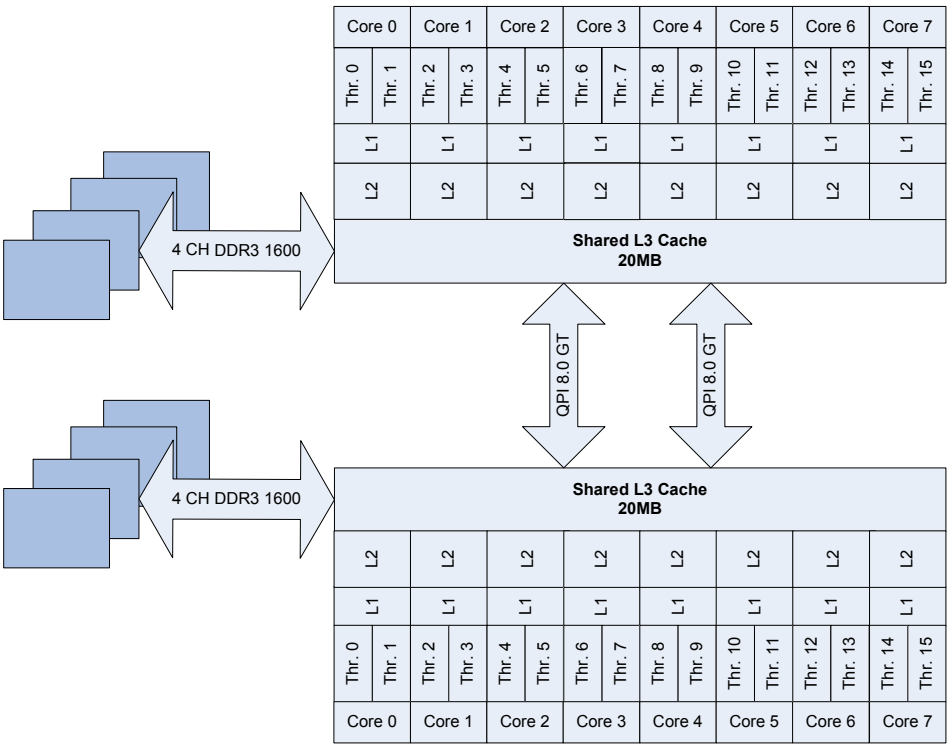

Abbildung 3: Intel Sandy Bridge NUMA-Architecture

For inter node communication, SuperMUC features an Infiniband network. Infiniband offers low latencies (1.2 us) at more then 6 GByte/s per second bandwidth. The overall network is organizes as a pruned fat-tree: 512 nodes form a so-called island with one high-end switch. 18 island are connected via a 1-level tree.

Other systems, like IBM's BlueGene or Cray's XK6 feature custom interconnects (which are routerless) in order to form a 2D/3D mesh. If a scientific application (e.g. based on regular grids or matrices) matches to this network only llowöverhead for communication is measured.

What is the peak performance of the system if a memory bound application is executed?

ANSWER:

SuperMUC features four channels per socket of DDR3-1600 RAM. On channel is able to deliver 12.8 GByte/s what results in a total bandwidth of 102.4 GByte/s.

Let's execute a matrix vector multiplication: For a $N \times N$ matrix we have to calculate N scalar products. For each scalar product we have to read $2N$ values which is have an amount of of $16N$ bytes and we are executing N FLOP. Therefore we are able to achieve 102.4/16 GFLOPS = 6.4 GFLOPS (1.8 % peak performance !!!)

Which hardware feature may help to improve this performance?

ANSWER:

Caches!

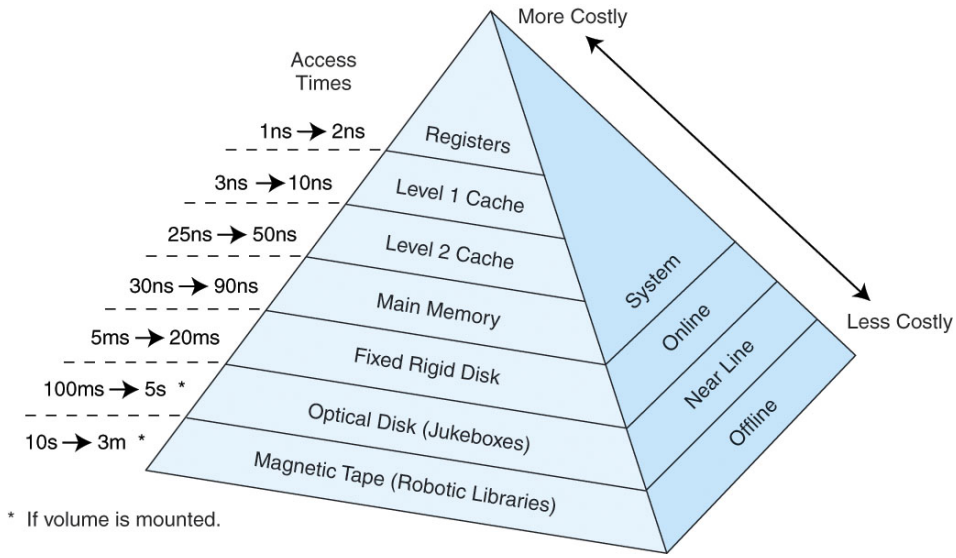

Abbildung 4: cache hierarchy

Let's assume we can store b of our matrix vector multiplication in the cache, we just have to stream A. This means for a scalar product we have to read $8N$ bytes which results in a performance of 12.8 GFLOPS.

$\Rightarrow$ In order to unleash the performance of today's chips cache awareness is a must.

3) Vectorization of Linked-Cells particle simulations

Recently C++ has established itself as one of the standard language for new development of HPC codes apart from FORTRAN and C. Please sketch the software layout of a linked-cells C++ implementation.

ANSWER:

Classes for: Particle, ParticleContainer, Cell, ForceCalculation.

Which problems do you face when it comes to low level kernels and data layout?

ANSWER:

Interactions are taking place on a particle basis. ⇒ ForceCalculation class gets two references and computes and updates velocities and forces for each particle. This leads to a so-called Array of Structures (AoS) which is performance-killing.

In order to speed-up calculations it's useful to flip this layout which leads to a cell-based only method, Fig. 5.

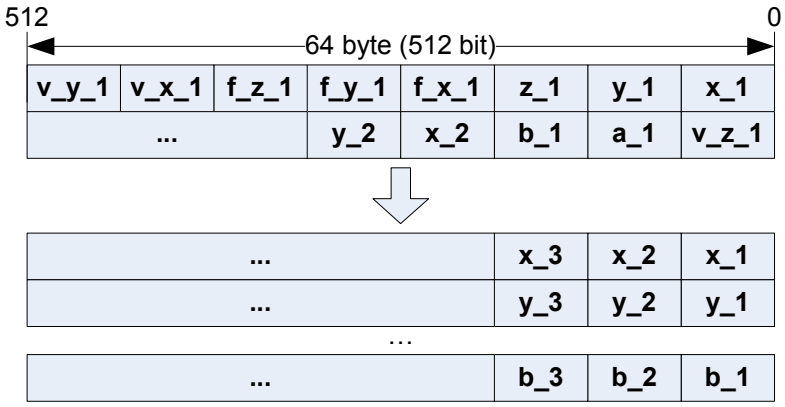

Abbildung 5: AoS to SoA conversion

Can you use vectorization in order to speed-up calculations?

Given a SoA-storage, we are able to vectorize the LJ-12-6 kernel across particle pairs. Please note that this approach scales with the number of particles per cell and does not require zero-padding or is limited to a vector-length of four as methods. In our case the only limitations are the number of particles per cell, the size of the cutoff-radius r_{cutoff} and cell-length r_c as these factors influence the vector load.

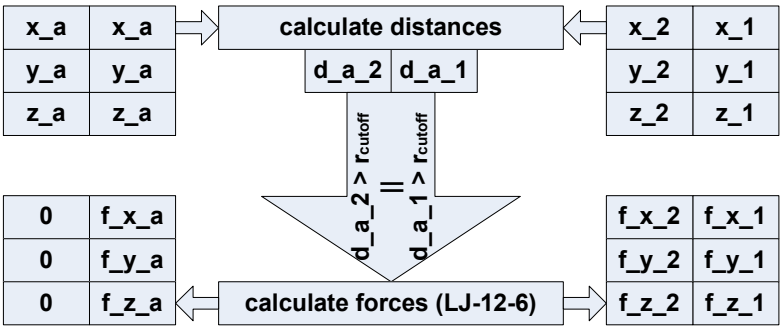

Abbildung 6: Vectorization of the LJ-12-6 force calculation kernel.

Figure 6 sketches the applied vectorization of the LJ-12-6 calculations. As we need to deal with double precision floating point numbers we can store a maximum of two elements per SSE register. The calculation is performed on particle pairs, therefore we broadcast-load (duplicating by vector-length) the required data of one particle in the first register (a) and the second register is filled by data of two different particles (1 and 2). With this approach we can theoretically reduce the number of operations by a factor of two. Since we are calculating two interactions within one step, we need to apply some pre- and post-processing, which can be performed by regular logical operations directly in hardware: the SSE/AVX instruction set offers compare, logical and sign-bit operations that can be used to decide if the force calculation should be initiated (at least one particle pair distance is smaller than r_{cutoff}, pre-processing) and if calculated results need to be zeroed by a mask (a particle pair distance is greater than r_{cutoff}). In case of SSE this should already gain a factor of two for small particle numbers. The lower performance bound is the scalar performance as these algorithms will not execute more instructions than necessary in the scalar case.

additional: 4) Complexity of the Barnes-Hut-Method

In the lecture the costs for the d-dimensional Barnes-Hut-Method were given as $O(\theta^{-d}NlogN)$. First derive the costs for the twodimensional Barnes-Hut-Method. Then explain descriptively (without proof), why the formula is also correct for 3d.

ANSWER:

The "outer loop" of the Barnes-Hut-algorithm iterates over all particles and calculates the force effective on every particle. As there are N particles in total, for each of those particles the costs have to be $O(\theta^{-d} \cdot log\ N)$. For the given complexity of $O(\theta^{-d} \cdot N \cdot log\ N)$ we assume that the tree is not degenerated, thus the tree

has $O(log\ N)$ levels. If we can show, that at each level the costs are $O(\theta^{-d})$, the formula is proven valid.

We start out from the twodimensional case. First we consider an arbitrary level and determine the number of nodes (i.e. particles of pseude-particles) we have to consider.Those are exactly the nodes for which the parent node didn't fulfill the θ-criterion. For the parent node it has to hold:

$\frac{diam}{|distance|} > \theta$. Thus it follows for the distance:

$$|distance| < \frac{diam}{\theta} \tag{1}$$

We will now try to determine an upper bound for the number of parent cells which fulfill that condition. Therefor we consider only one coordinate axis first.

Not considering the absolute value, (1) can be rewritten as $-\frac{diam}{\theta} < distance < \frac{diam}{\theta}$. Thus the "area" of the distance is $2 \cdot \frac{diam}{\theta}$. As one node coveres an area with the diameter $diam$ there can be only

$$\frac{2 \cdot \frac{diam}{\theta}}{diam} = \frac{2}{\theta}$$

nodes next to each other, for which the θ-criterion does not hold. In the second axis there are just as many. Thus there are

$$\left(\frac{2}{\theta}\right)^2 = 4 \cdot \theta^{-2}$$

nodes in total, which don't fulfill the θ-criterion. Those are the parent nodes we have to consider at the current level. That number still has to be multiplied by 4, what doesn't change the order.

Thus the maximum computational costs on each level are $O(\theta^{-2})$. As there area $log\ N$ levels in total and N particles, the total computational costs sum up to $O(\theta^{-2} \cdot N \cdot log\ N)$.

For three dimensions it is easy to see that there are not $c \cdot \theta^{-2}$ any more for which the θ-criterion is fulfilled, but only $c \cdot \theta^{-3}$, as we are dealing now with a cubic area.

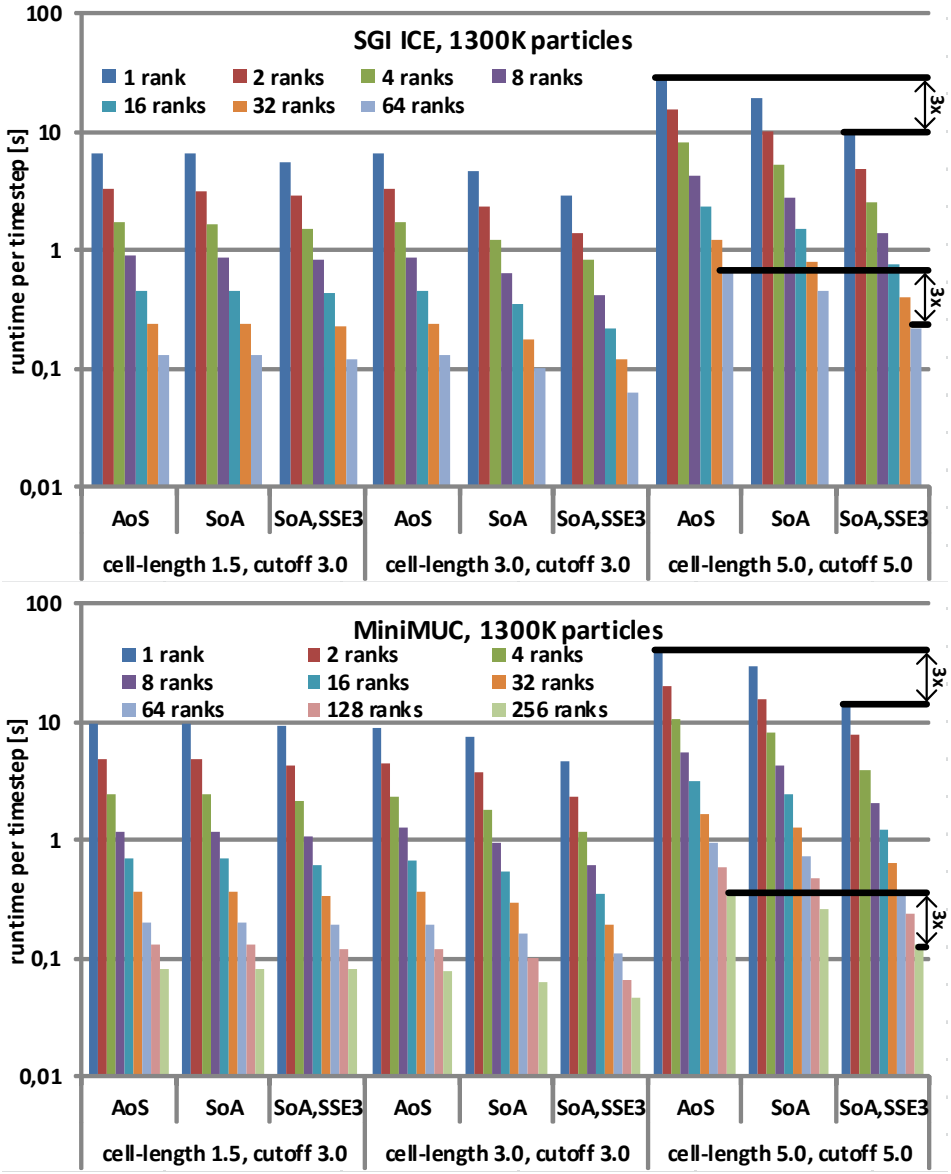

Abbildung 7: Runtimes in seconds for a scenario with 1300k particles on both clusters, one MPI rank per core.