Sun UltraSPARC T2
First Tests

Georg Hager
Gerhard Wellein
Thomas Zeiser
Michael Meier

HPC Services, RRZE
SunDay RRZE, 2007/11/06
Outline

- Sun UltraSPARC T2 basics
- Lattice Boltzmann Method (LBM) benchmarks
- RRZE benchmark suite (very preliminary)
- STREAM performance numbers
  - How to get good data streaming
- Thanks to Sun and RWTH Aachen for granting access to EA T2 system
UltraSPARC T2: Server on a Chip

- 8 SPARC V9 cores @ 1.2–1.4GHz
  - 8 threads per core
  - 2 execution pipelines per core
  - 1 instruction/cycle per pipeline
  - 1 FPU per core
  - 1 SPU (crypto) per core
  - 4 MB, 16-way, 8-bank L2$

- 4 FB-DIMM DRAM controllers
- 2.5 GHz x 8 PCI-Express interface
- 2 x 10 Gb on-chip Ethernet
- Technology: TI 65nm
- Die size: 342mm$^2$
- Power: < 95 W (nominal)
- On-Chip Encryption: DES, 3DES, AES, RC4, SHA1, SHA256, MD5, RSA 2048, ECC, CRC32
Lattice Boltzmann Method

Evaluating single node performance with an OpenMP Kernel
The Lattice Boltzmann Method (LBM)

- Evolved from “cellular automata models”
- Physical basis: the Boltzmann equation

\[
\frac{\partial f}{\partial t} + \mathbf{\xi} \frac{\partial f}{\partial \mathbf{x}} + \mathbf{K} \frac{\partial f}{\partial \mathbf{\xi}} = Q(f, f)
\]

1) approximation of the collision process by the BGK relaxation
2) physical discretisation → „velocity discrete Boltzmann equation“
3) numerical discretisation of spatial and temporal derivatives

⇒ explicit lattice Boltzmann equation

\[
f_i(\mathbf{x} + \mathbf{c}_i, t + 1) - f_i(\mathbf{x}, t) = -\frac{1}{\tau} (f_i - f_i^{eq})
\]

- satisfies the incompressible Navier Stokes equations with 2nd order accuracy
- numerical and computational advantages
Optimal data access patterns:
Basic implementation strategy: LIJK layout

double precision \( F(0:18,0:xMax+1,0:yMax+1,0:zMax+1,0:1) \)
do \( z=1,zMax \)
do \( y=1,yMax \)
do \( x=1,xMax \)
if( fluidcell(x,y,z) ) then
\( \text{LOAD } F(0:18,x,y,z,t) \) \[
\text{Relaxation (complex computations)} \]
SAVE \( F(0,x,y,z,t+1) \)
SAVE \( F(1,x+1,y+1,z,t+1) \)
SAVE \( F(2,x,y+1,z,t+1) \)
SAVE \( F(3,x-1,y+1,z,t+1) \)
…
SAVE \( F(18,x,y-1,z-1,t+1) \)
endif
enddo
enddo
enddo

LD & ST
19 Cachelines

Collide Step

Stream Step

If cache line of store operation is not in cache it must be loaded first (RFO)!

#load operations: \( 19\times xMax\times yMax\times zMax + 19\times xMax\times yMax\times zMax \)

#store operations: \( 19\times xMax\times yMax\times zMax \)

Assuming full use of each cache line!

06.11.2007 hpc@rrze.uni-erlangen.de

UltraSPARC 12 benchmarks
double precision  \( f(0:x_{\text{Max}}+1,0:y_{\text{Max}}+1,0:z_{\text{Max}}+1,0:18,0:1) \)

!$OMP PARALLEL DO PRIVATE(Y,X, …) SCHEDULE(RUNTIME)

do  z=1,z_{\text{Max}}
  do  y=1,y_{\text{Max}}
    do  x=1,x_{\text{Max}}
      if( fluidcell(x,y,z) ) then
        LOAD \( f(x,y,z,0:18,t) \)
        Relaxation (complex computations)
        SAVE \( f(x,y,z,0,t+1) \)
        SAVE \( f(x+1,y+1,z,1,t+1) \)
        ...
        SAVE \( f(x,y-1,z-1,18,t+1) \)
      endif
    enddo
  enddo
enddo

Optimal data access patterns:

Basic implementation strategy: IJKL layout & OpenMP

Collide

Stream

#load operations: \( 19 \times x_{\text{Max}} \times y_{\text{Max}} \times z_{\text{Max}} \) + \( 19 \times x_{\text{Max}} \times y_{\text{Max}} \times z_{\text{Max}} \)

#store operations: \( 19 \times x_{\text{Max}} \times y_{\text{Max}} \times z_{\text{Max}} \)
Optimal data access patterns:
Basic implementation strategy: VECTOR & OpenMP

```fortran
double precision f(0:((xMax+1)*(yMax+1)*(0:zMax+1)),0:18,0:1)
 !$OMP PARALLEL DO PRIVATE(...) SCHEDULE(RUNTIME)
 do m=1,(zMax*yMax*xMax)
   if( fluidcell(m) ) then
     LOAD f(m, 0:18,t)
     Relaxation (complex computations)
     SAVE f(m , 0,t+1)
     SAVE f(m-(xMax+2)-(xMax+2)*(yMax+2), 18,t+1)
   endif
 enddo
enddo

Pitfall on ccNUMA systems: Initialization is still of IJKL style…

  !$OMP PARALLEL DO PRIVATE(Y,X,...) SCHEDULE(RUNTIME)
  do z=1,zMax
    do y=1,yMax; do x=1,xMax
      f(x ,y ,z , 0:18,t)=...
    enddo; enddo
  enddo
enddo
```

06.11.2007  hpc@rrze.uni-erlangen.de  UltraSPARC T2 benchmarks
Optimal data access patterns:  
Setting up a simple performance model for LBM

- Crossover between cache and memory bound computations:  
  \[ 2 \times 19 \times x_{\text{Max}}^3 \times 8\text{Byte} \sim L2/L3 \text{ cache size} \rightarrow x_{\text{Max}} \sim 14-15 \text{ (for 1 MB cache)} \]

- Data must be transferred from and to main memory in each time step:  
  - Assumption: full use of each cache line loaded

- Data to be transferred for a single fluid cell update:  
  \[(2+1)\times 19 \times 8 \text{ Byte} \rightarrow 456 \text{ Bytes/(cell update)}\]

- Max. performance: Million Lattice Site Updates per second  
  \[ M_{\text{MLUPs}} = \frac{\text{Memory Bandwidth [MByte/s]}}{(456 \text{ Bytes/cell update})} \]

- Good approximation: MemoryBandwidth = “STREAM TRIADS”

- \#Floating Point Operations varies: 150 – 200 per fluid cell update  
  \[ 5 \text{ MLUPs} \leftrightarrow 1 \text{ GFlop/s} \]
Optimal data access patterns:
Setting up a simple performance model for LBM

- So far we could not convince the Intel compiler to use “Non-Temporal stores”
- Single Core estimates

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel Xeon/Nocona</td>
<td>6.4</td>
<td>32</td>
<td>3,000</td>
<td>6.6</td>
<td>4.5-5.0</td>
</tr>
<tr>
<td>Intel Xeon/Woodcrest</td>
<td>12.0</td>
<td>60</td>
<td>4,200</td>
<td>9.2</td>
<td>7.0-8.0</td>
</tr>
<tr>
<td>AMD Opteron875</td>
<td>4.4</td>
<td>22</td>
<td>3,500</td>
<td>7.7</td>
<td>4.5-5.0</td>
</tr>
<tr>
<td>Intel Montecito</td>
<td>6.4</td>
<td>32</td>
<td>6,100</td>
<td>13.4</td>
<td>9.0-10.0</td>
</tr>
<tr>
<td>NEC SX8</td>
<td>16</td>
<td>80</td>
<td>~60,000</td>
<td>~200</td>
<td>~65</td>
</tr>
</tbody>
</table>

Simple model provides a good approximation

Vector-performance not limited by memory bandwidth!

06.11.2007 hpc@rrze.uni-erlangen.de
UltraSPARC T2 benchmarks
LBM
Performance – Single node: 2-socket Woodcrest

LBMKernel: woody (HP DL140G3)
OMP_NUM_THREADS=8 ifort9.1 (-fast -openmp; SPLITLOOPS)

FluidMLUPS

2-socket Intel Xeon DualCore (Woodcrest)
3.0 GHz; Greencreek (SF=ON)
LBMKernel: *thunder* (HP DL585G2)

OMP_NUM_THREADS=8 ifort9.1 (-W -O3 -openmp; SPLITLOOPS)

4-socket AMD Opteron DualCore
2.8 GHz; socket F
LBM
Performance – Single node: 4-socket Tigerton

LBMKernel: \texttt{tigerton1} (Intel EA platform)

OMP\_NUM\_THREADS=8 ifort9.1 (-fast -openmp; SPLITLOOPS)

\begin{itemize}
\item IJKL - OMP\_SCHEDULE=static
\item VECTOR - OMP\_SCHEDULE=static
\item VECTOR - OMP\_SCHEDULE=dynamic,1000
\end{itemize}

4-socket Intel Xeon QuadCore
2.93 GHz; SF=ON
Tigerton/Caneland
1 thread/chip

\begin{figure}
\centering
\includegraphics[width=\textwidth]{chart.png}
\caption{FluidMLUPS vs. domain size for different configurations.}
\end{figure}
LBM
Performance – Single node: 1-socket Niagara2

LBMKernel: Niagara2 (SUN EA / RWTH Aachen)
-openmp -fast -xtarget=ultraT2 -xcache=8/16/4:4096/64/16 -m64 -xarch=sparcvis2 (studio12)

1-socket: 8 cores / 64 threads max.
SUN EA (status: alpha)
Measurements: 22./23.8.2007
LBM
Performance – Compare them all

LBMKernel

- Woody: 4 threads; 4 cores; 2 sockets
- Tigerton: 8 threads; 16 cores; 4 sockets
- Thunder: 8 threads; 8 cores; 4 sockets
- Niagara2: 32 threads; 8 cores; 1 socket

FluidMLUPS vs. domain size
RRZE Benchmark Suite

- Standardized, easy-to-use benchmark suite for use in evaluations, procurements etc.
- Several benchmark codes (applications and low-level) under a common Makefile construct
- Customizing one single include file and typing “make” does the trick
- Performance data collected as simple numbers (higher is better) with timestamps for easy reference

- Benchmarks used in previous procurement:
  AMBER, EXX, IMD, OAK3D, TRATS, PIO, TRIAD

- Additional benchmarks in the suite: Kette, DMRG
Benchmark results
RRZE Benchmark Suite: EXX & AMBER8

- **EXX**
  - Quantum Chemistry package developed at Theoretical Chemistry, U Erlangen
  - Calculation of structural and electronic properties of periodic systems (solids, slabs, wires)
  - Using (time-dependent) Density Functional Theory (DFT)
  - Performance dominated by FFT operations (FFTW)
  - Largely cache-bound
  - Fortran 90 and MPI

- **AMBER8 (pmemd)**
  - Widely used commercial MD package
  - Distributed-memory FFT and force field calculations
  - Benchmark case (HCD): HPr:CCpa Tetramer dynamics
  - Largely cache-bound
  - Fortran 77 and MPI
Benchmark results
RRZE Benchmark Suite

- **IMD**
  - Molecular dynamics package developed at U Stuttgart
  - Used at FAU for calculation of defect dynamics in solids
  - Weakly dependent on memory bandwidth
  - C and MPI

- **TRATS**
  - Production Lattice-Boltzmann CFD code developed at Institute for Fluid Dynamics, U Erlangen
  - Memory-bound on standard microcomputers, compute-bound on NEC SX (code balance: $\approx 0.4$ Words/Flop)
  - Fortran 90 and MPI
  - Also known as BEST
  - RRZE built a small kernel to evaluate optimization approaches & single CPU/node characteristics (LBMkernel)
RRZE Benchmark Suite

- **OAK3D**
  - Physics code developed at Theoretical Physics, U Erlangen
  - Simulates the dynamics of exotic (superheavy) nuclei via time-dependent Hartree-Fock (TDHF)
  - Photoabsorption, e\(^{-}\) capture, fusion, fission
  - Taylor expansion of time evolution and predictor-corrector step
  - Uses FFT for calculating derivatives
  - Performance dominated by small-size FFTs and dense matrix-vector operations
  - Some memory bandwidth required, benefits from large caches
  - Fortran 90 and MPI
1-node shootout RRZE Benchmark suite

Woody: 3GHz HP DL140G3 (4C)
## UltraSPARC T2 benchmarks

For Reference: Memory bandwidth
Optimized version of STREAM: Tigerton (Intel EA)

<table>
<thead>
<tr>
<th>4 sockets (SF=ON) QuadCore@2.93 GHz</th>
<th>COPY [MB/s]</th>
<th>SCALE [MB/s]</th>
<th>ADD [MB/s]</th>
<th>TRIAD [MB/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 thread (1 Chip; 1 FSB)</td>
<td>2753</td>
<td>2798</td>
<td>2650</td>
<td>2641</td>
</tr>
<tr>
<td>4 threads (2 Chips; 1 FSB)</td>
<td>2855</td>
<td>2855</td>
<td>2680</td>
<td>2680</td>
</tr>
<tr>
<td>4 threads (4 Chips; 2 FSB)</td>
<td>5656</td>
<td>5678</td>
<td>5360</td>
<td>5350</td>
</tr>
<tr>
<td>4 threads (4 Chips; 4 FSB)</td>
<td>8345</td>
<td>8442</td>
<td>10260</td>
<td>10264</td>
</tr>
<tr>
<td>8 threads (8 chips; 4 FSB)</td>
<td>8686</td>
<td>8601</td>
<td>10735</td>
<td>10715</td>
</tr>
</tbody>
</table>

**AMD Opteron DC 2.8 GHz 4 sockets / DDR667 / 8 threads**

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Best out of 10</td>
<td>17035</td>
<td>17154</td>
<td>17591</td>
<td>17625</td>
</tr>
<tr>
<td>Worst out of 10</td>
<td>14982</td>
<td>15088</td>
<td>15363</td>
<td>15379</td>
</tr>
</tbody>
</table>
Memory bandwidth
Stream: Niagara2 (SUN EA – status: alpha)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>(ARRAY=32<em>1024</em>1024) COMMON=ON</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16 threads (pinned) (offset=0)</td>
<td>13818</td>
<td>9681</td>
<td>5001</td>
<td>5158</td>
</tr>
<tr>
<td>16 threads (pinned) (offset=1)</td>
<td>9302</td>
<td>8612</td>
<td>4784</td>
<td>5134</td>
</tr>
<tr>
<td>16 threads (pinned) (offset=512)</td>
<td>13124</td>
<td>9629</td>
<td>5267</td>
<td>5237</td>
</tr>
<tr>
<td>16 threads (pinned) (offset=512+16)</td>
<td>12375</td>
<td>10383</td>
<td>12400</td>
<td>12129</td>
</tr>
</tbody>
</table>

Observations

• Physical addresses mapping to memory controller uncontrollable?

• RFO on write misses → “effective bandwidth” increases by 3/2 (4/3) for COPY & SCALE (ADD & TRIAD)
Memory bandwidth – stream
Niagara2 (SUN EA – status: alpha; SW upgrade)
Memory bandwidth – stream
Niagara2 (SUN EA – status: alpha; SW upgrade)
Memory bandwidth – stream
Niagara2 (SUN EA – status: alpha; SW upgrade)

COMMON A,B,C - Triads: A=B+s*C
DOUBLE A(SIZE+OFFSET),... (SIZE=2^5)

Graph showing memory bandwidth over offset with different thread counts.
Memory bandwidth – stream
Niagara2 (SUN EA – status: alpha; SW upgrade)

COMMON A,B,C - Add: C=A+B
DOUBLE A(SIZE+OFFSET).... (SIZE=2^25)
Memory bandwidth – stream
Niagara2 (SUN EA – status: alpha; SW upgrade)
Memory bandwidth – stream
Niagara2 (SUN EA – status: alpha; SW upgrade)

COMMON A,B,C - Copy: C=A
DOUBLE A(SIZE+OFFSET),... (SIZE=2^{25})

Compiler using memcpy() w/ block stores if alignment is right (RFO avoided)
Vector triad \( A(1:N) = B(\cdot) + C(\cdot) \ast D(\cdot) \)

- Observation
  - Mutual alignment of arrays has high impact on performance
  - Slight change in \( N \) can have big effect

- How can access patterns be optimized independent of \( N \)?
  - Alignment of array start addresses to page boundaries is bad
    - Severe conflicts
  - Different OpenMP chunks (“segments”) can be aligned and shifted vs. alignment boundary
    - ameliorates bottlenecks, but still erratic numbers
Vector triad $A(1:N)=B(:) + C(:) \times D(:)$

- Arrays $A$, $B$, $C$, $D$ can be aligned to page boundaries and shifted w.r.t each other
  - Address bits 8:7 determine mapping to memory controller
  - Bit 6 chooses L2 bank per controller
  - Best result for shift = 128 (see next slide)

Array $N$ shifted by $N \times 128$ bytes
Schönauer vector triad \( A(1:N) = B(:) + C(:) \times D(:, \cdot) \)

32 threads

![Graph showing performance metrics](image)

- **Use of block stores (VIS) to eliminate RFO?**
- **L2 limit**

17.3 GB/s not counting RFO (21.6 w/ RFO)

---

06.11.2007

hpc@rrze.uni-erlangen.de

UltraSPARC T2 benchmarks

31
Conclusions

- Interesting massively threaded architecture
- Watch data alignment and aliasing issues
- Sensible measure for HPC: Performance per box per $