Performance Evaluation of Current HPC Architectures Using Low-Level and Application Benchmarks

G. Hager, H. Stengel, T. Zeiser, G. Wellein

Regionales Rechenzentrum Erlangen (RRZE)
Friedrich-Alexander-Universität Erlangen-Nürnberg

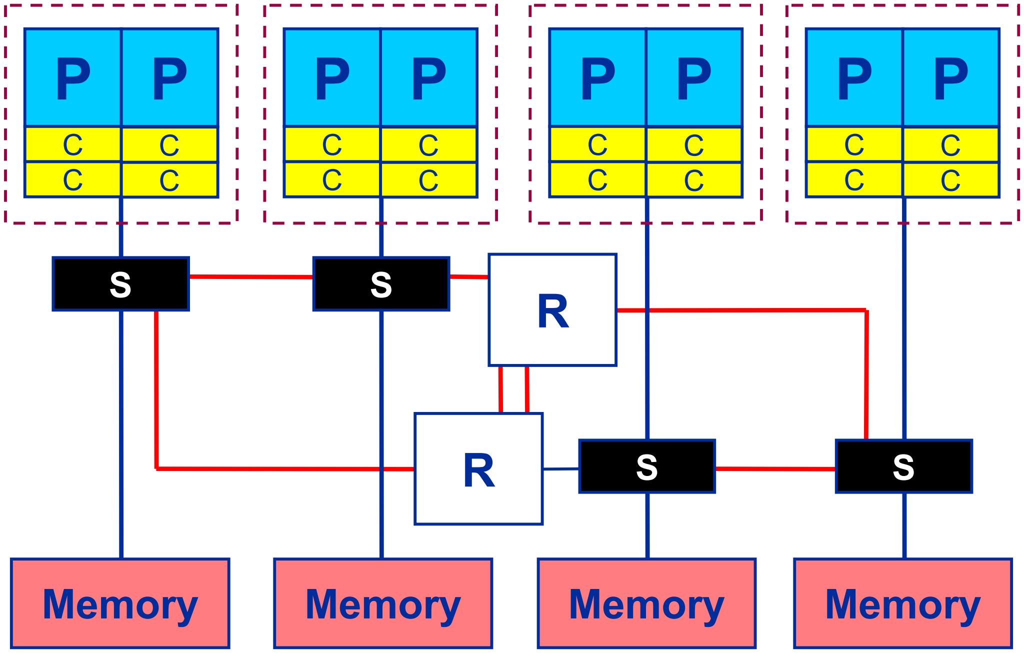
HLRB2/KONWIHR Results & Review Workshop
LRZ, 2007/12/03

Outline

- Architectures under consideration
  - SGI Altix 4700 (HLRB II)
  - RRZE Woodcrest cluster
  - RRZE “Port Townsend” cluster
  - Sun UltraSPARC T2 (“Niagara 2”)
  - … and a few others
- Low-level benchmarks
  - Some comments on the Pallas/Intel MPI PingPong test
  - Peculiarities of the new Sun UltraSPARC T2 processor
- Application benchmark suite RZBENCH
  - Short description of the benchmarks
  - Presentation of results
    - core, socket, node, cluster
- Conclusions

Processor and cluster architectures:
HLRB II

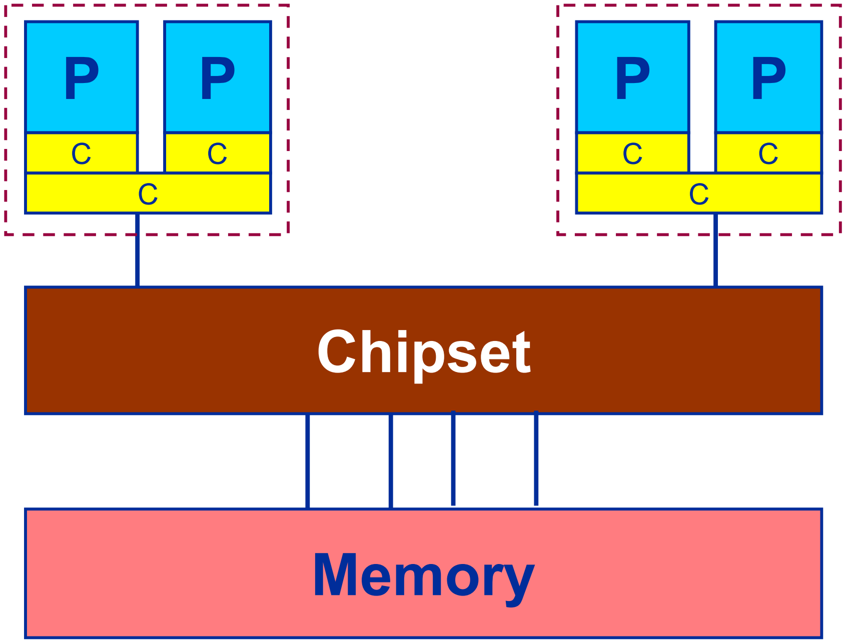
- Dual core Itanium 2 (“Montecito”) @ 1.6 GHz, 9MB L3
  - 6-way superscalar in-order, 2xFMA/cycle (4 FLOPs)
  - 8.5 GB/s memory BW per socket
- ccNUMA nodes w/ 512 cores each
  - High density and bandwidth variants available
  - NUMALink 4 connections (3.2 GB/s per direction)
  - Fat tree network



- “Constellation” system

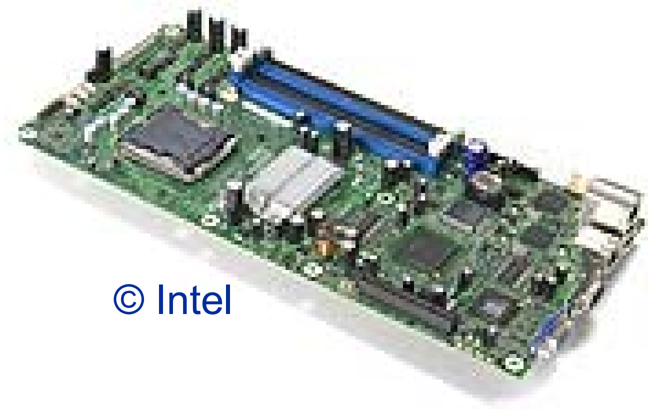
Processor and cluster architectures: Woody – RRZE Woodcrest cluster

- Intel Xeon 5160 dual core 3 GHz CPU (“Core 2 Duo”)
  - 64 kB L1 per core
  - 4 MB shared L2 per socket
- 4 FLOPs (1xMULPD, 1xADDPD) per cycle
- 10.6 GB/s nominal FSB BW per socket
  - “Greencreek” chipset w/ Snoop Filter
  - HP DL140G3 node
  - Aggregate STREAM ≈ 6.1GB/s
- “DDR-x” IB network
  - 216 (288) port Voltaire switch
  - DDR inside lineboard
  - SDR on spine switches



Processor and cluster architectures: "Port Townsend" cluster at RRZE

http://www.intel.com/design/servers/boards/s3000PT/index.htm



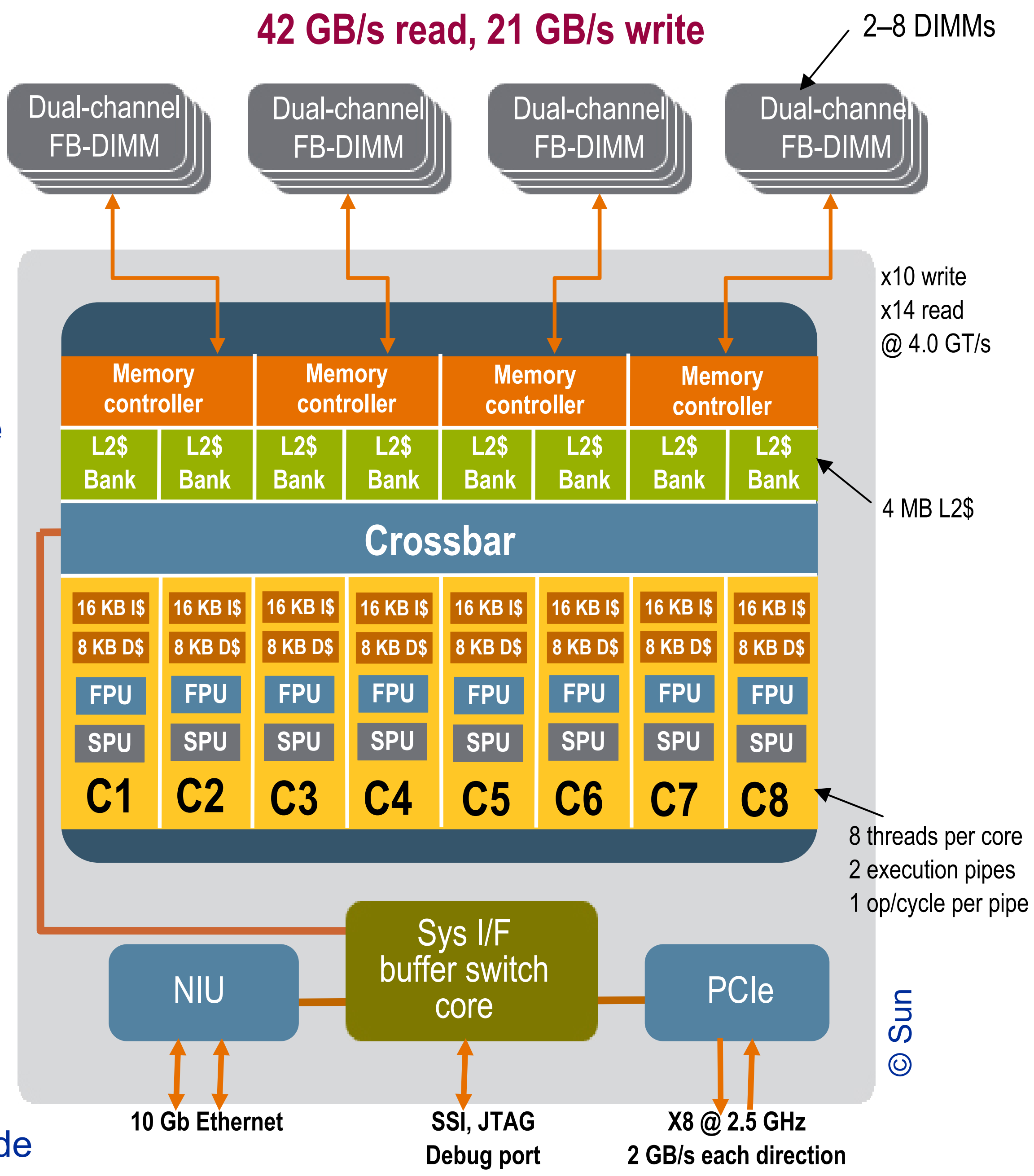
© Intel

- Optimized for
  - MPI apps with high memory and/or MPI bandwidth requirements!
- Not optimized for
  - Maximum LINPACK/$

- Intel S3000PT board
  - 1 socket: Intel Xeon30XX series
  - 2 boards/ 1 U
  - FSB1066 – unbuffered DDR2
  - 1 PCIe-8X
  - 2 SATA ports
  - Intel AMT
- RRZE S3000PT Cluster (09/2007)
  - 66 compute nodes
    - 2,66 GHz Xeon 3070 (dual core)
    - 4 GB memory (DDR2-533)
  - 72-port Flextronics IB DDR-Switch (max. 144 ports)
  - Delivered by transtec

UltraSPARC T2: Server on a Chip

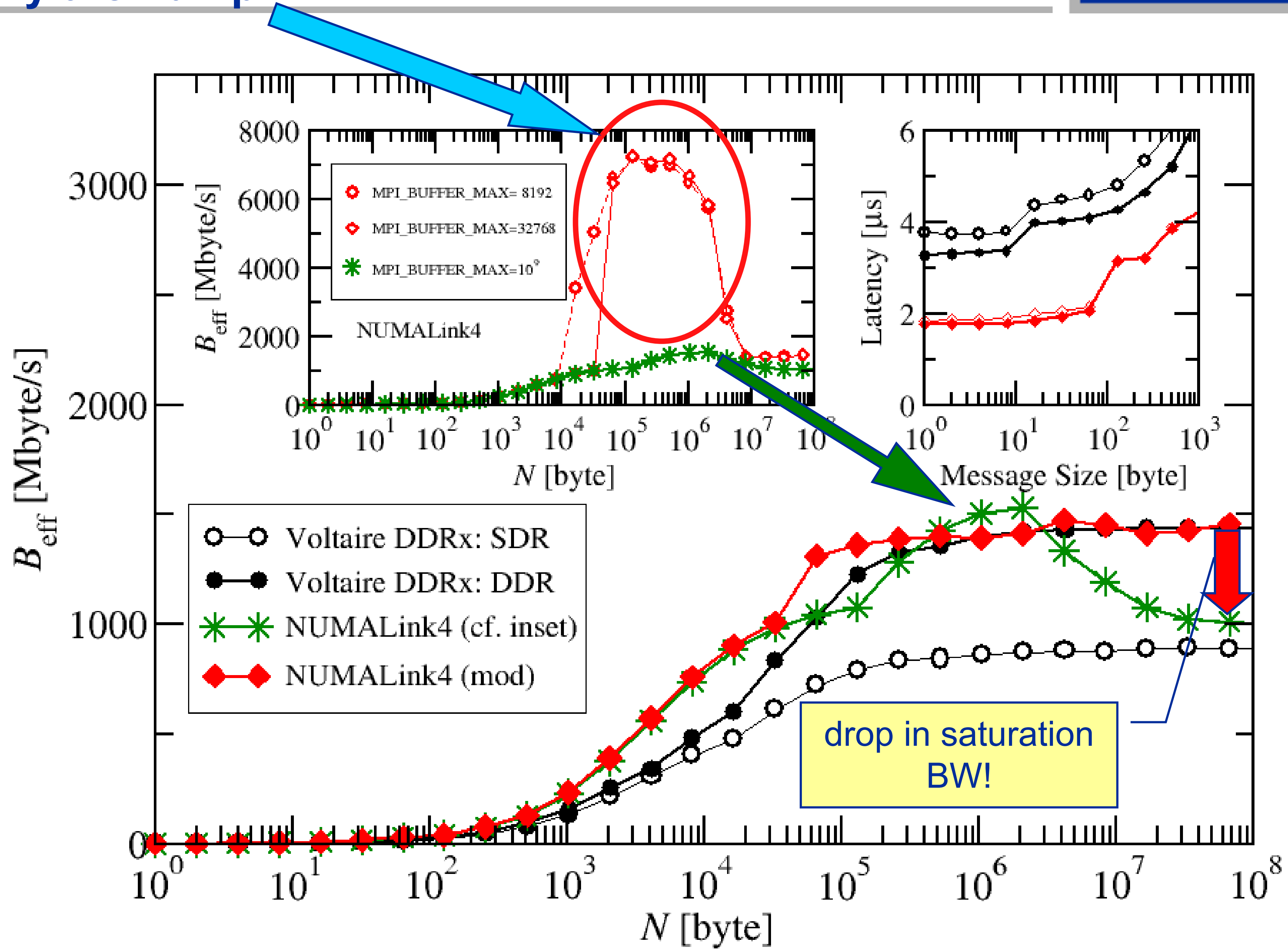
- 8 SPARC V9 cores @ 1.2–1.4GHz
  - 8 threads per core
  - 2 execution pipelines per core
  - 1 instruction/cycle per pipeline
  - 1 FPU per core
  - 1 SPU (crypto) per core
  - 4 MB, 16-way, 8-bank L2$
- 4 FB-DIMM DRAM controllers
- 2.5 GHz x 8 PCI-Express interface
- 2 x 10 Gb on-chip Ethernet
- Technology: TI 65nm
- Die size: 342mm^2
- Power: < 95 W (nominal)
- On-Chip Encryption:
  DES, 3DES, AES, RC4, SHA1, SHA256, MD5, RSA 2048, ECC, CRC32



© Sun

The Pallas/Intel MPI PingPong test:
Some comments regarding shared memory systems

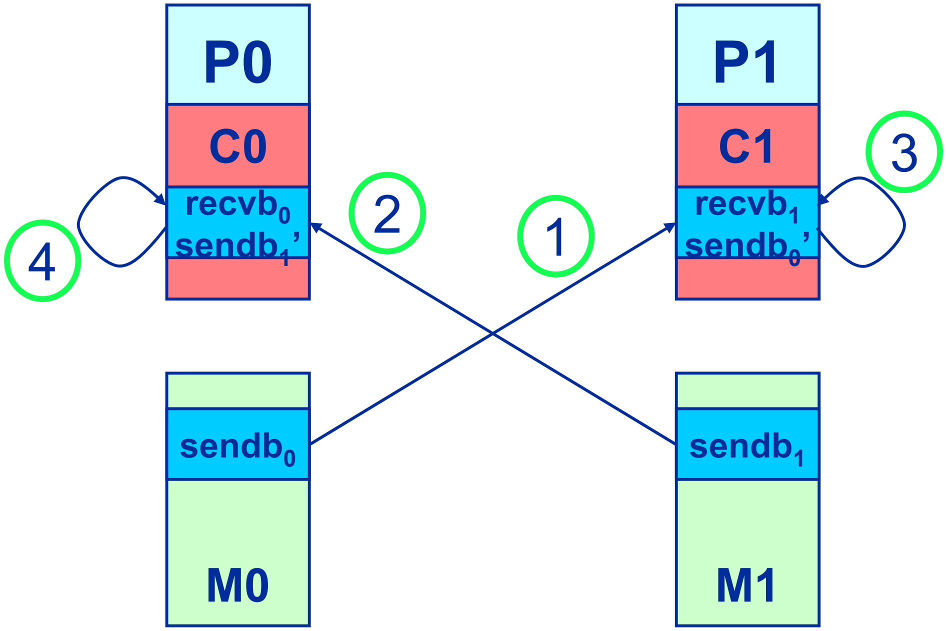
PingPong on Altix: Why the hump?



drop in saturation BW!

PingPong on Altix:
Single Copy Transfers for N>MPI_BUFFER_MAX

- Chain of events for single copy `MPI_Send` on ccNUMA:
  1. First ping: P1 copies sendb$_0$ to recvb$_1$
  2. First Pong: P0 copies sendb$_1$ to recvb$_0$
  3. Second Ping: P1 performs in-cache copy on unmodified recvb$_1$
  4. Second Pong: P0 performs in-cache copy on unmodified recvb$_0$
  5. …



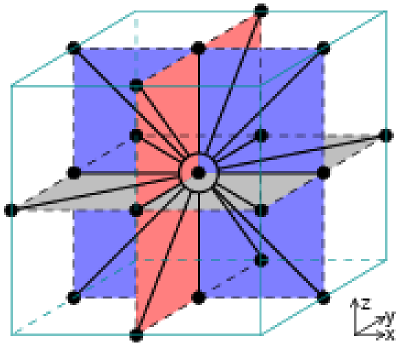
- This effect can not be exploited for real applications!
  - Nevertheless, results are reported on HPCC site
- Remedy: Add second Ping-Pong with sendb$_i$ and recvb$_i$ interchanged

Sun UltraSPARC T2:
Memory access peculiarities

Lattice Boltzmann Kernel:

IJKL layout & OpenMP

```
double precision f(0:xMax+1,0:yMax+1,0:zMax+1,0:18,0:1)
!$OMP PARALLEL DO PRIVATE(Y,X,…) SCHEDULE(RUNTIME)
do z=1,zMax
   do y=1,yMax
      do x=1,xMax
           if( fluidcell(x,y,z) ) then
Collide {   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)
Stream  {
            …
            SAVE f(x  ,y-1,z-1, 18,t+1)
           endif
      enddo
   enddo
enddo
```

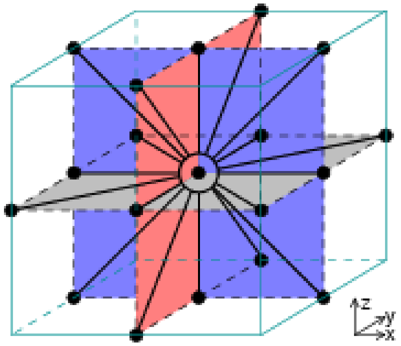


#load operations: **19***xMax*yMax*zMax
+ **19***xMax*yMax*zMax

#store operations: **19***xMax*yMax*zMax

Lattice Boltzmann Kernel:
VECTOR & OpenMP

```
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
```



Pitfall on ccNUMA systems: Observe proper parallel initialization!

```
!$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
```

Optimal data access patterns:

Setting up a simple performance model for LBM

- Crossover between cache and memory bound computations:
  *2 *19 *xMax^3 * 8Byte ~ L2/L3 cache size* → xMax ~ 14-15 (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**)*19*8 Byte → 456 Bytes/(cell update)
  - Max. performance: Million Lattice Site Updates per second

    MaxMLUPs=
    MemoryBandwidth [MByte/s] / (456 Bytes/cell update)
  - Good approximation: **MemoryBandwidth** = “STREAM TRIADS”

- *#Floating Point Operations varies: 150 – 200 per fluid cell update*
  5 MLUPs ⟷ 1 GFlop/s

Optimal data access patterns:

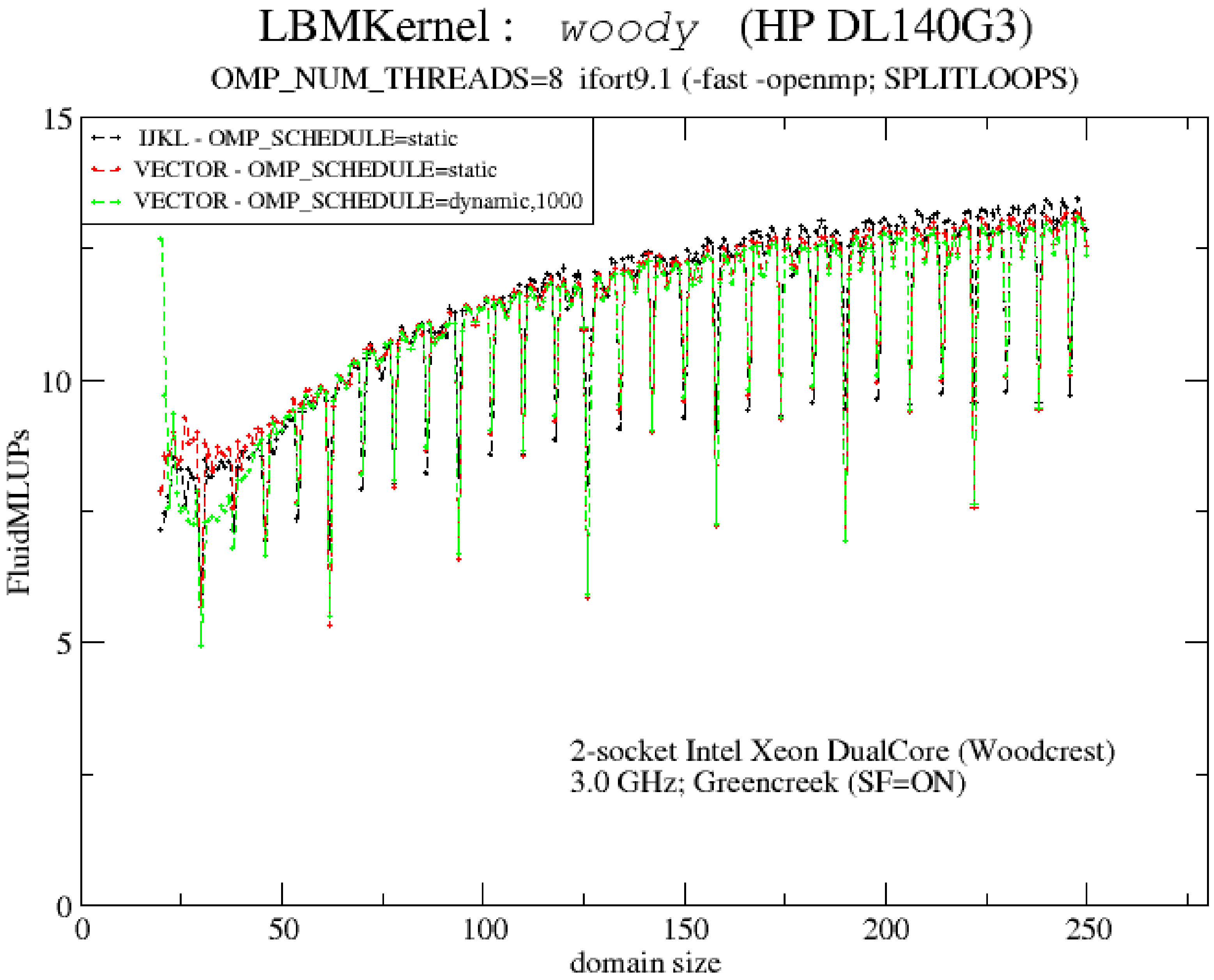
Setting up a simple performance model for LBM

- **Intel compiler refuses to use "Non-Temporal stores"**
- **Single Core estimates**

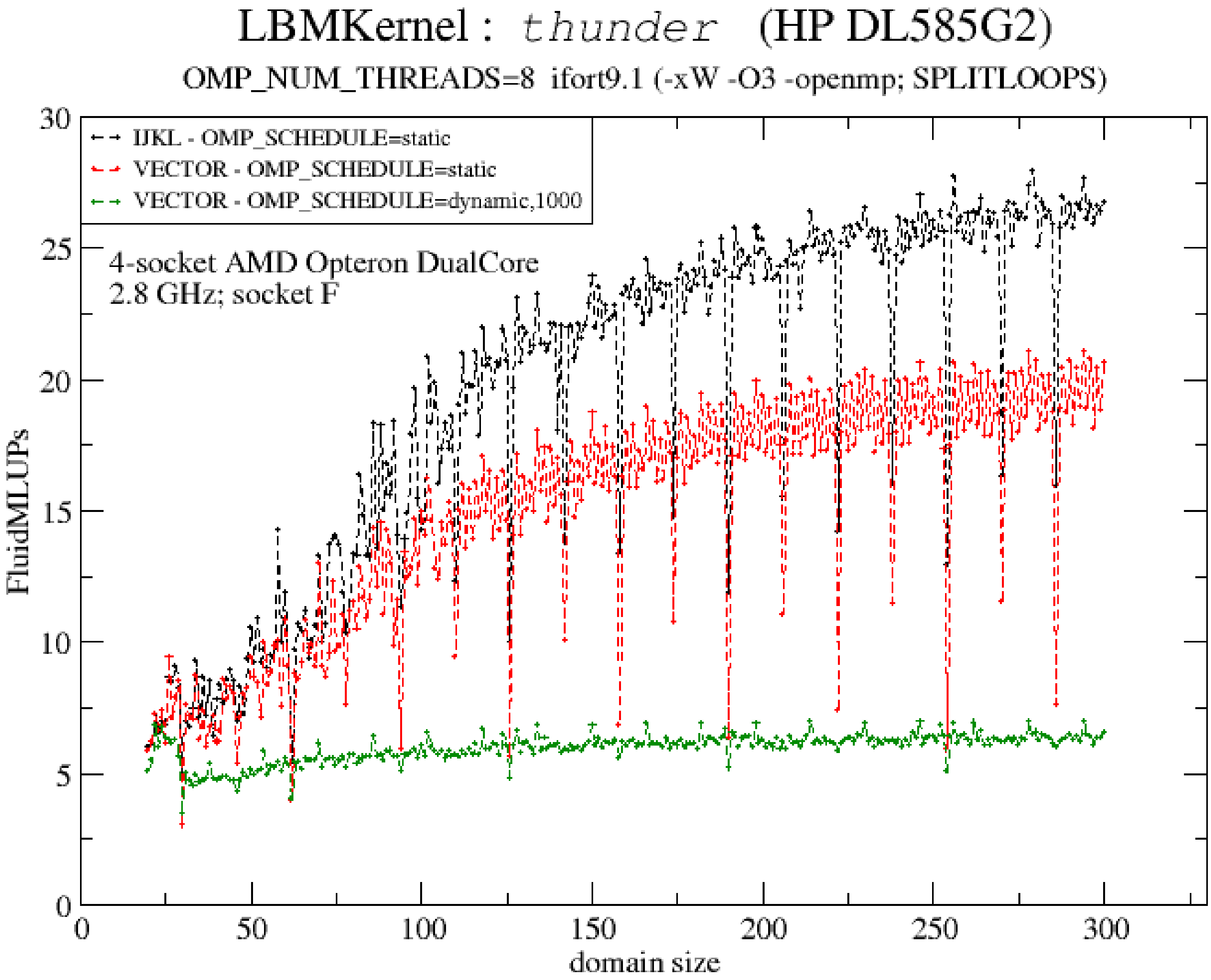
	Peak Perf. [GFlop/s]	Max. MLUPS	STREAM [MByte/s]	Max. MLUPS	**Max. Measure**
Intel Xeon/Nocona	6.4	32	3,000	**6.6**	**4.5-5.0**
Intel Xeon/Woodcrest	12.0	60	4,200	**9.2**	**7.0-8.0**
AMD Opteron875	4.4	22	3,500	**7.7**	**4.5-5.0**
Intel Montecito	6.4	32	6,100	**13.4**	**9.0-10.0**
NEC SX8	16	**80**	~60,000	**~200**	**~65**

- **Simple model provides a good approximation**
- **Vector performance not limited by memory bandwidth!**

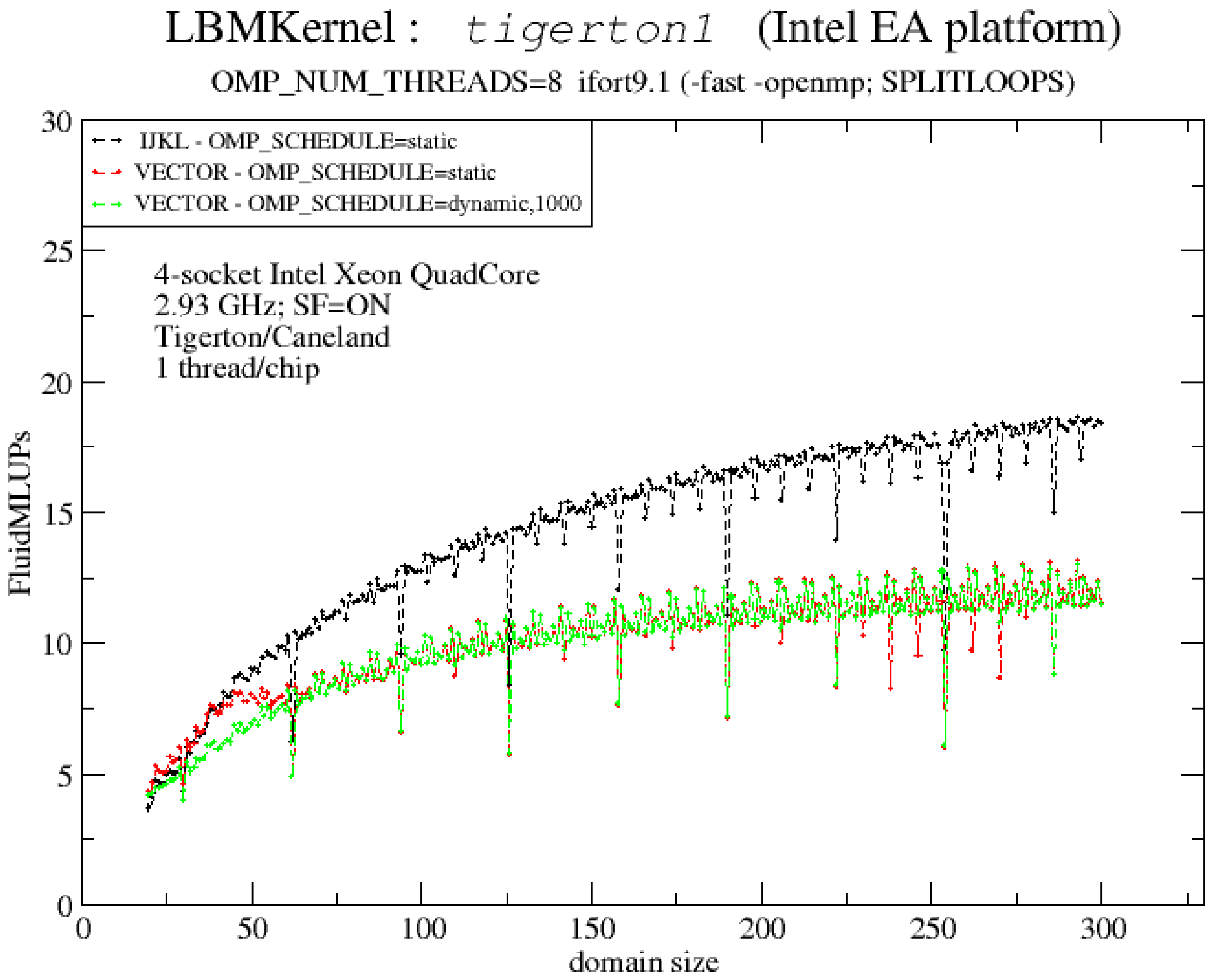
LBM
Performance – Single node: 2-socket Woodcrest



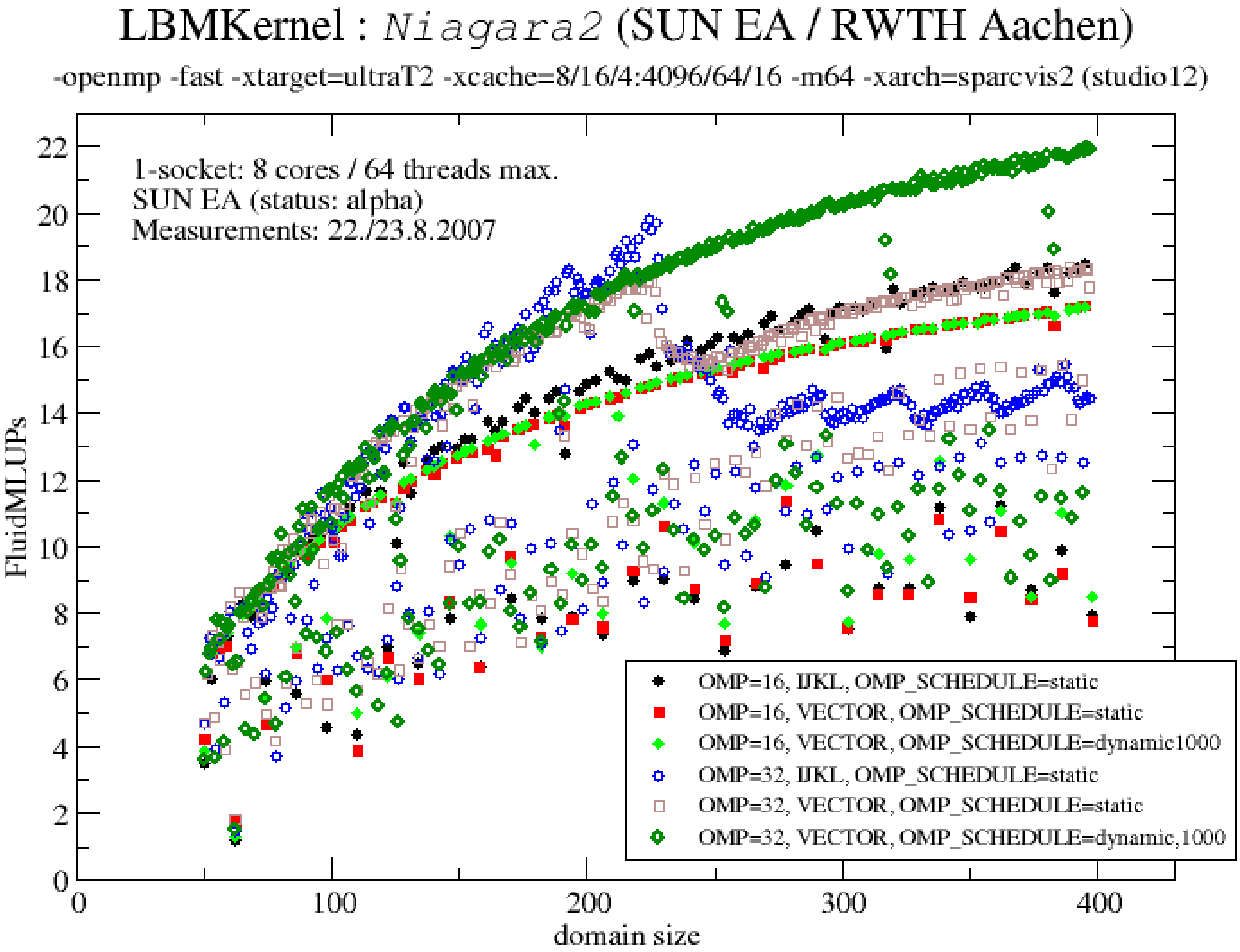
LBM
Performance – Single node: 4-socket F Opteron DC



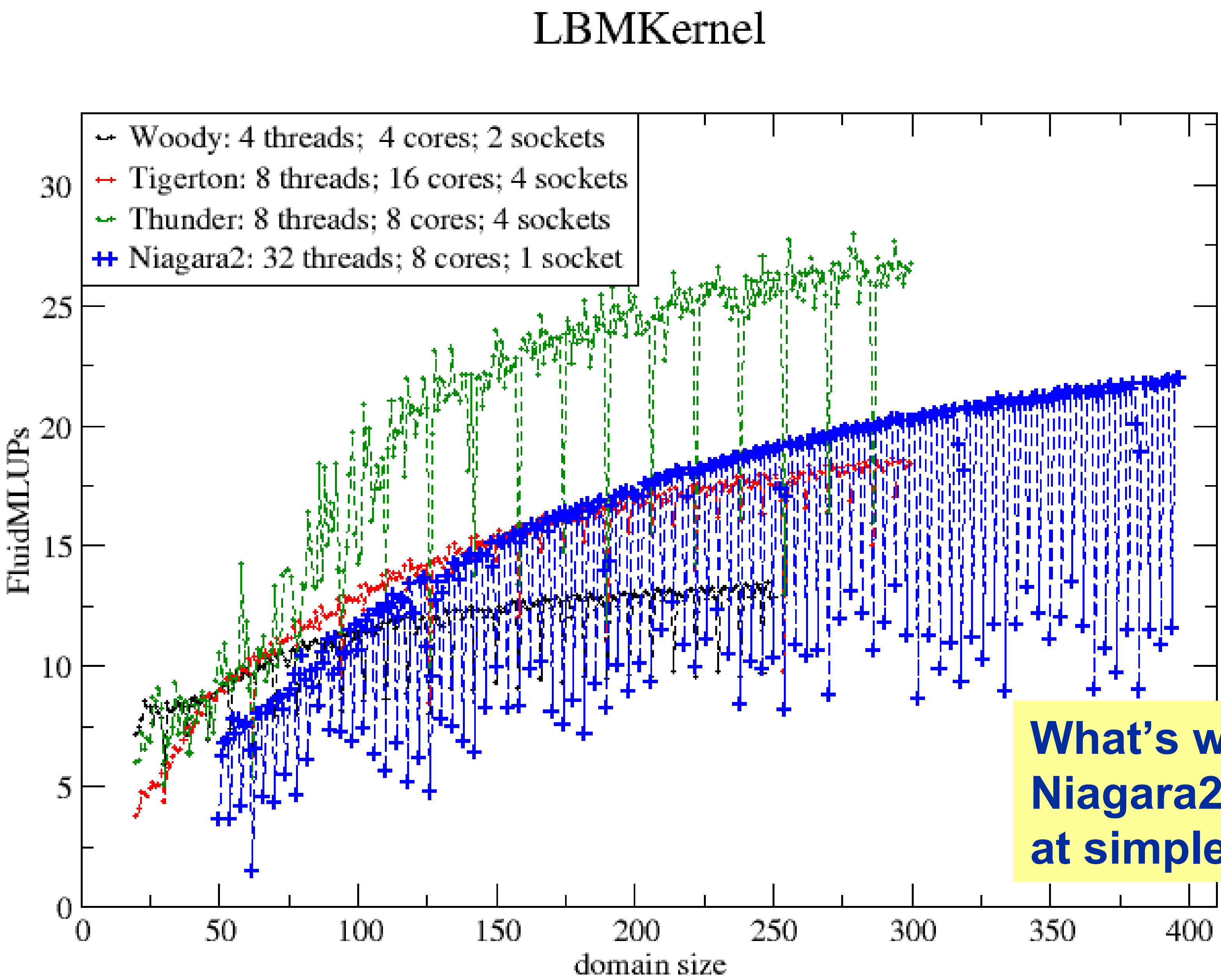
LBM
Performance – Single node: 4-socket Tigerton



LBM
Performance – Single node: 1-socket Niagara2

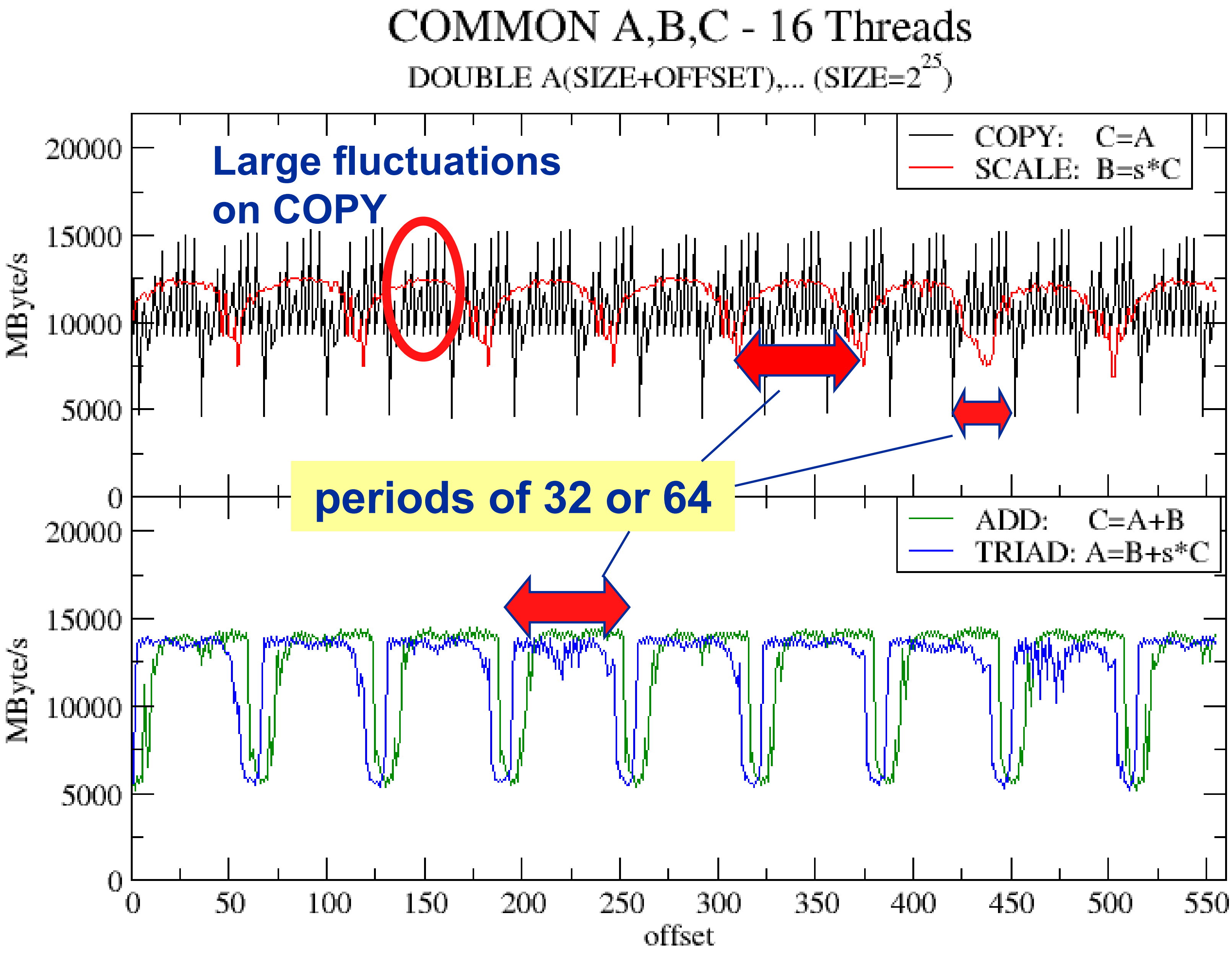


LBM
Performance – Compare them all



What's wrong with Niagara2? → look at simpler patterns!

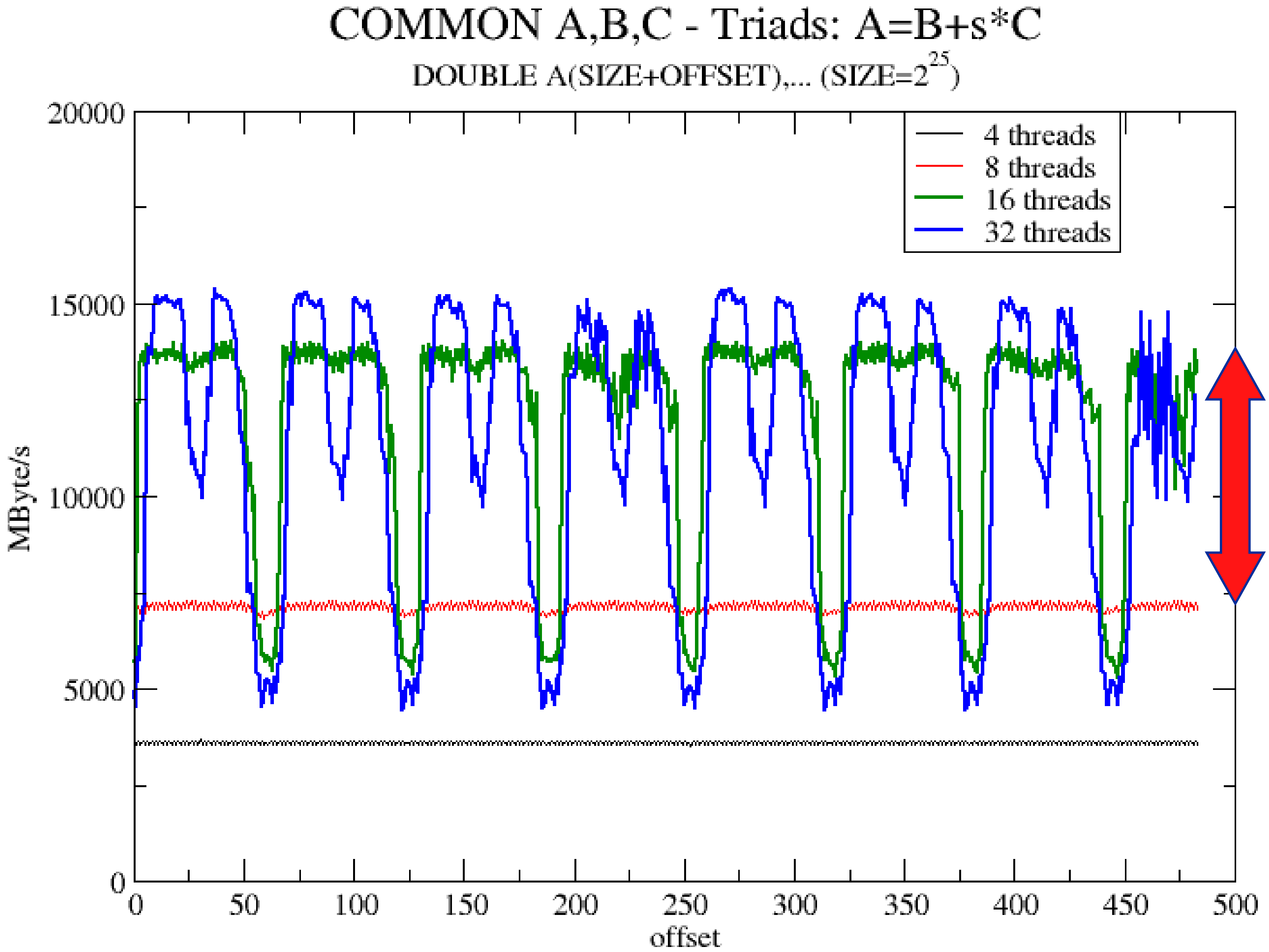
Memory bandwidth – STREAM on Niagara 2
Performance vs. array offset



Large fluctuations on COPY

periods of 32 or 64

Memory bandwidth – STREAM on Niagara2
TRIAD for different numbers of threads



8 threads cannot saturate memory bus!

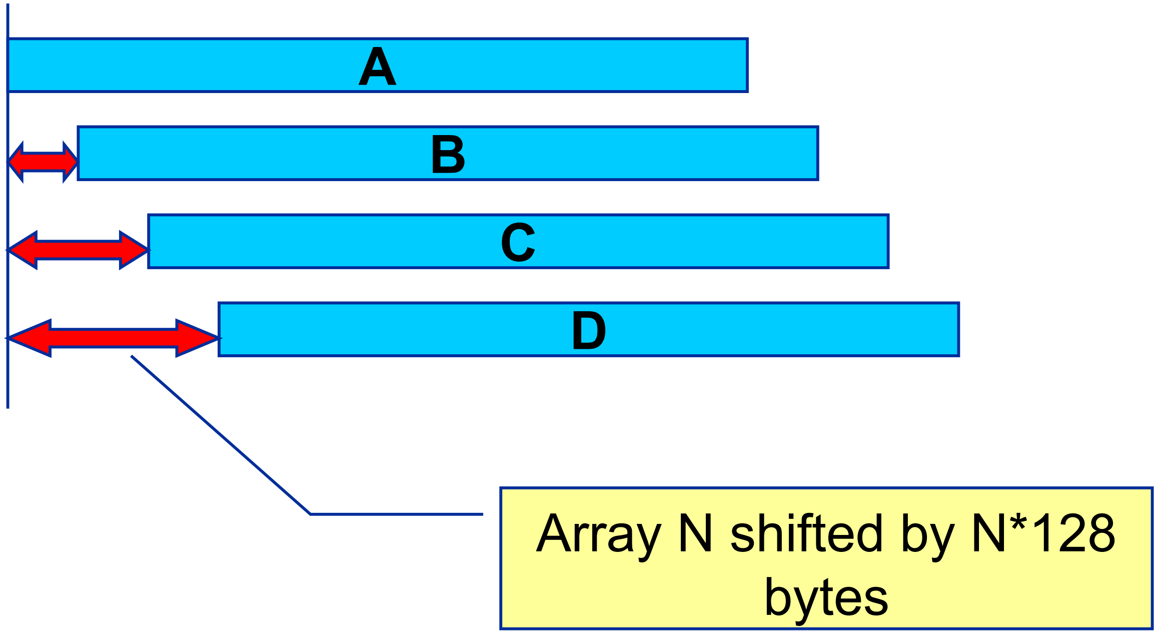
Vector triad A(1:N)=B(:) +C(:)*D(:)

- 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”) should be aligned and shifted vs. alignment boundary
    - ameliorates bottlenecks, but still erratic numbers

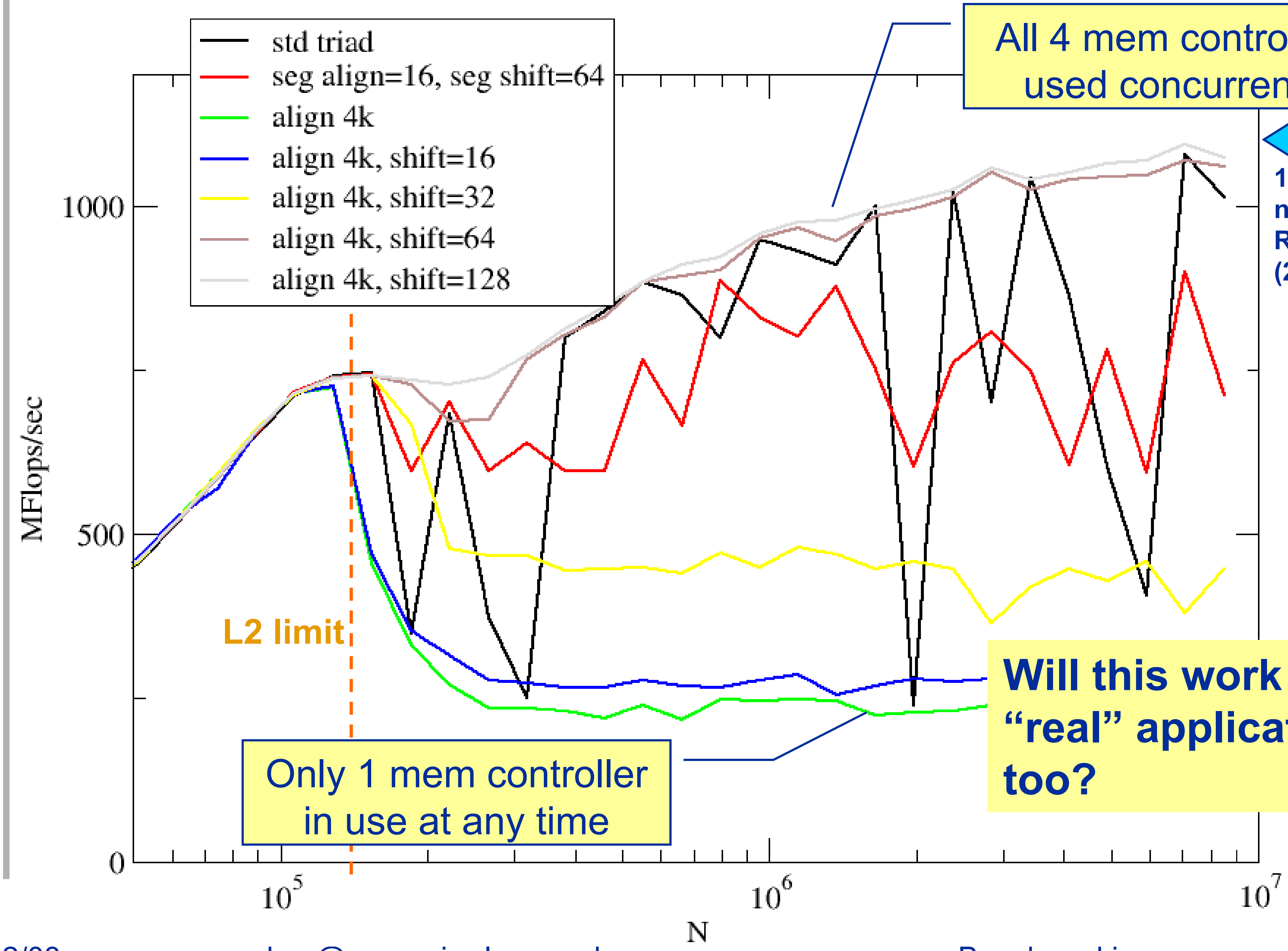
Vector triad A(1:N)=B(:) +C(:)*D(:)

- 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*128 bytes

Schönauer vector triad A(1:N)=B(:) +C(:)*D(:)
32 threads



All 4 mem controllers used concurrently

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

Only 1 mem controller in use at any time

Will this work with “real” applications, too?

Test case:
2D Jacobi heat conduction solver

- **Standard 5-point stencil, two arrays:**

```
#pragma omp parallel for ...
for(int i=1; i<size-1; i++)
  for(int j=1; j<size-1; j++)
    T[t1][i][j] = (T[t0][i-1][j] + T[t0][i+1][j]
                 + T[t0][i][j-1] + T[t0][i][j+1])*0.25;
```

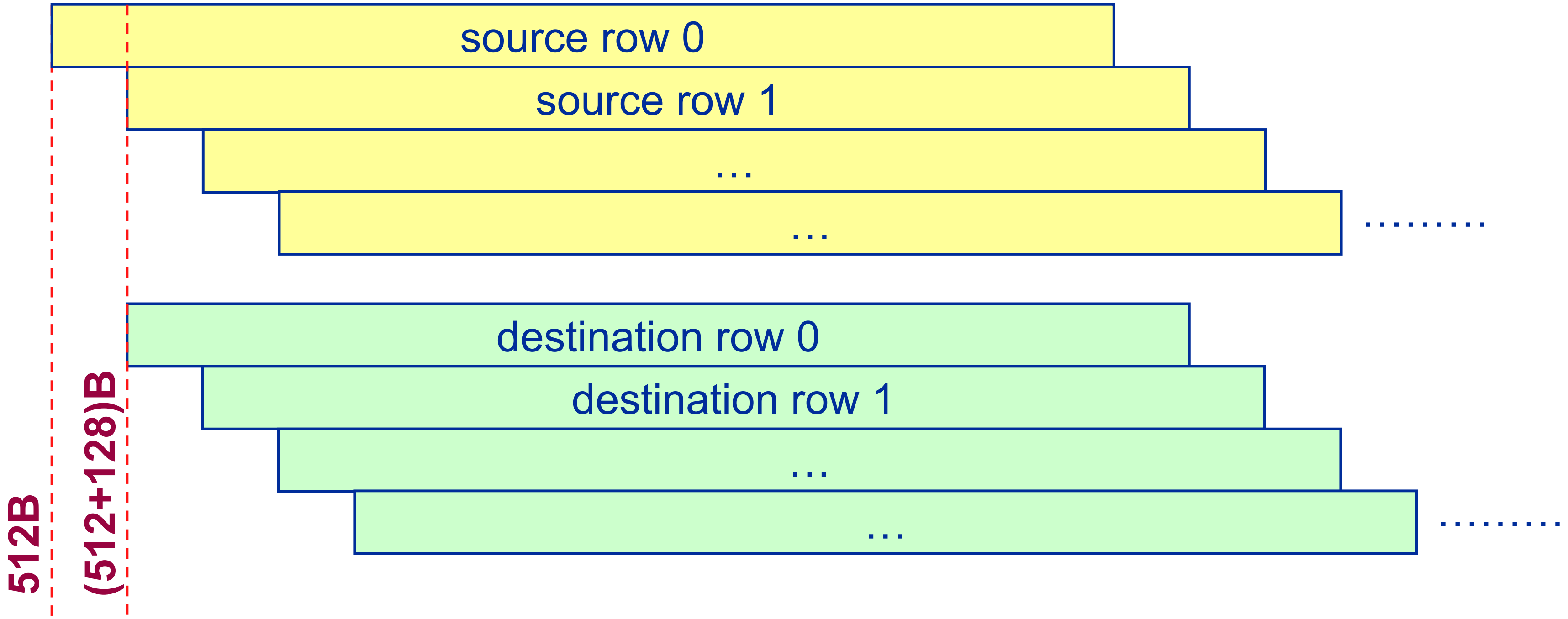
- **Align rows to appropriate boundaries e.g. by padding or separate allocation:**

```
for(int j=1; j<size-1; j++)
  dest_line[j] = (source_above[j] + source_under[j]
                + source_line[j-1]+ source_line[j+1])*0.25;
```

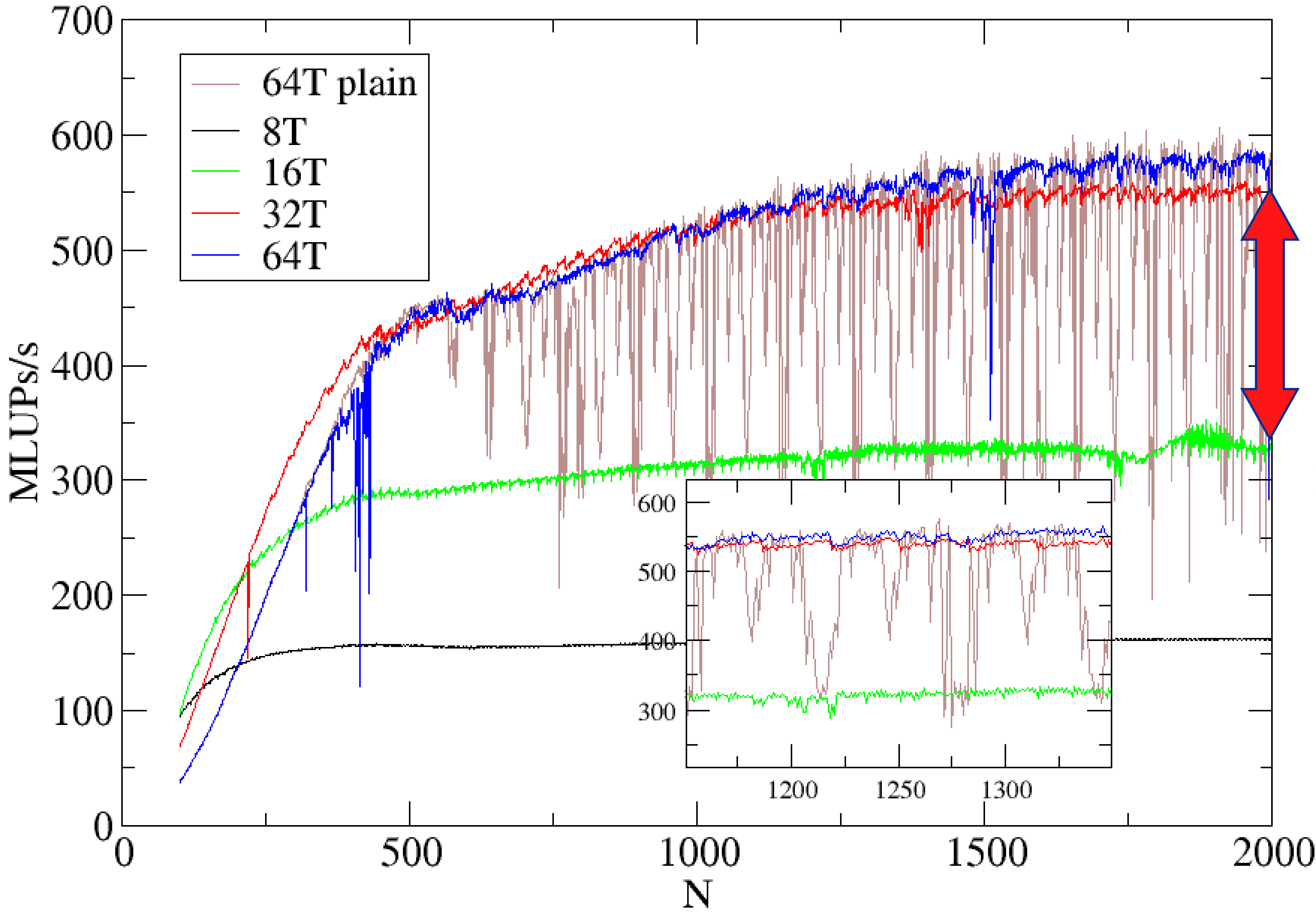
  - **Outer loop still parallelized, but now also responsible for address calculations**

2D Jacobi heat conduction solver: Optimal data alignment in Niagara 2

- **Align rows to 512 byte boundaries**
  - So, at first, each row starts at memory controller 0
- **Shift row *i* by 128 bytes**
  - So rows *i* and *i*+1 hit different memory controllers
- **Offset "source" and "destination" arrays by 128 bytes**
  - So the read and write streams of a single OpenMP thread hit different memory controllers

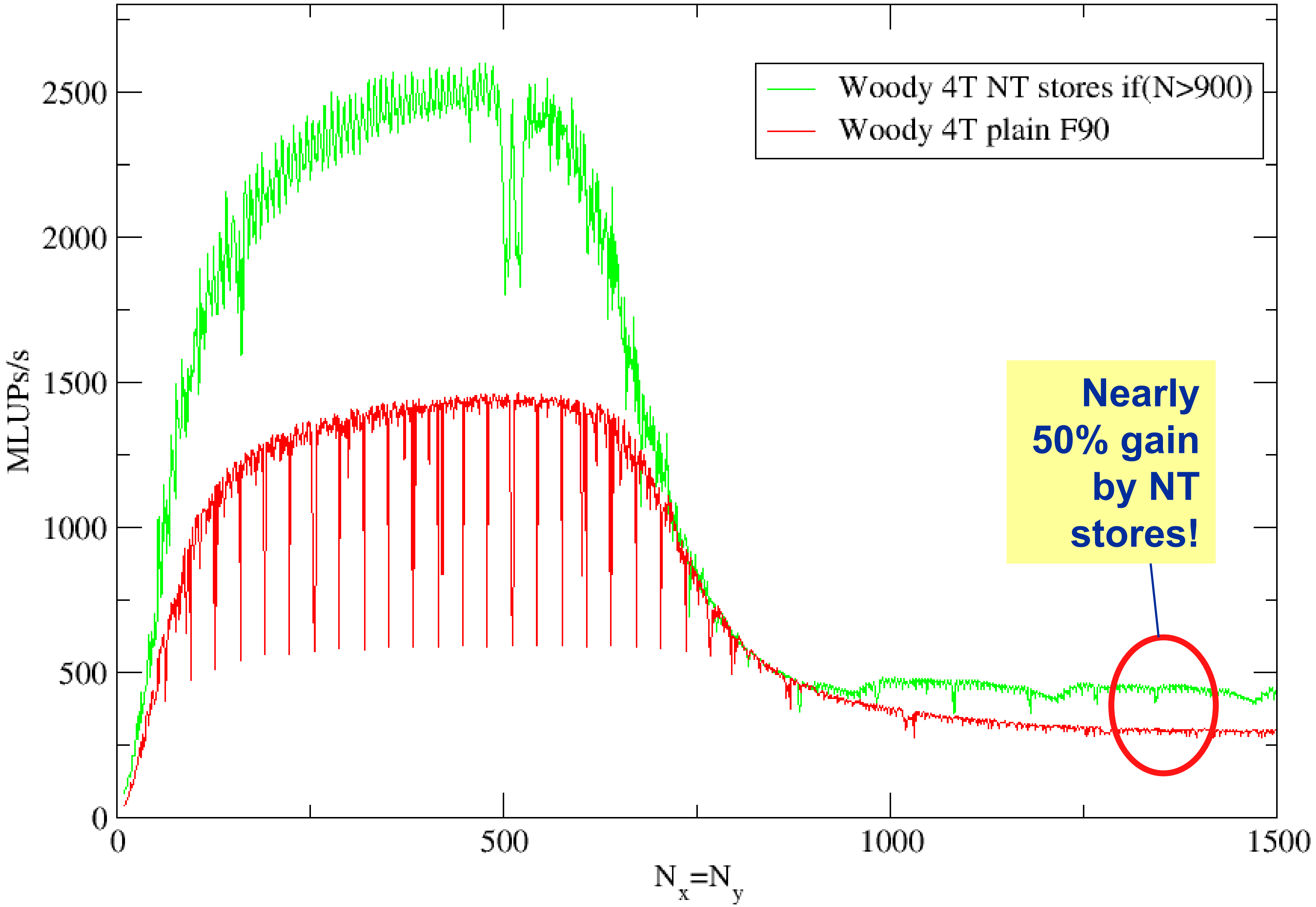


Doing it right on Niagara 2:
Optimal alignment and (static,1) OMP scheduling



32 threads needed

How about standard architectures?
E.g., nontemporal stores require proper alignment:



Nearly 50% gain by NT stores!

RZBENCH
Comparing systems using an integrated benchmark suite

RZBENCH

- Integrated suite containing
  - (currently) 9 application codes from FAU, more on the way
  - 2 low-level benchmarks
  - Centralized makefile construct
  - Run scripts for all benchmarks
  - Automatic result extraction and correctness checking (where appropriate)
  - One-stop configuration (single `make.inc.$ARCH`)
- Used in previous procurement
- Used in (EA) system evaluations

Short descriptions of the benchmarks: 5 used for most tests – cache friendly group

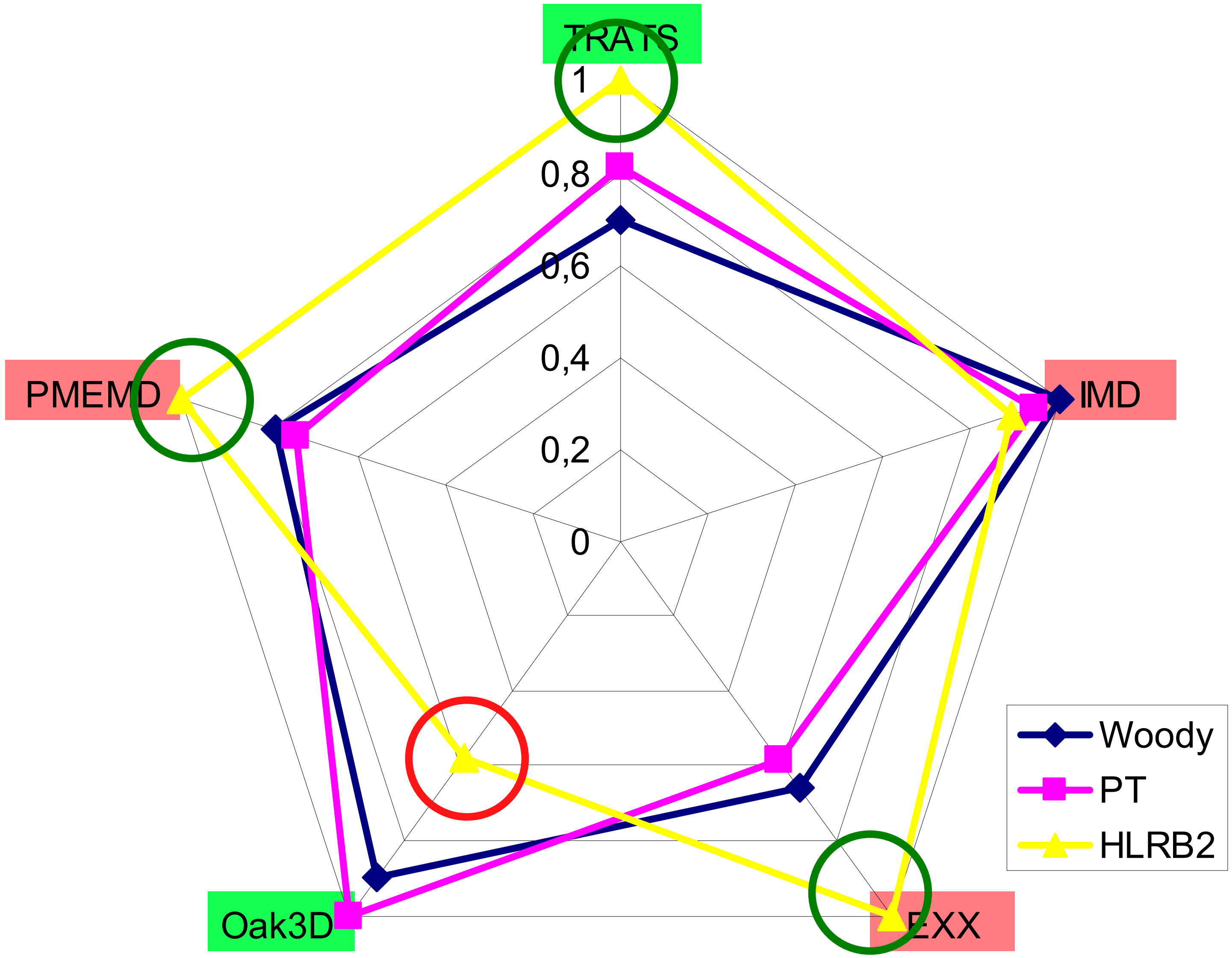
- **AMBER8/PMEMD**
  - Commercial MD package
  - Benchmark case largely cache bound
- **EXX**
  - FAU, theoretical chemistry
  - Time-dependent DFT
  - FFTW dominates performance
  - Largely cache bound
- **IMD**
  - MD package from U Stuttgart
  - FAU: used by materials scientists for simulation of defect dynamics
  - Weak dependence on memory BW

Short descriptions of the benchmarks: 5 used for most tests – bandwidth hungry group

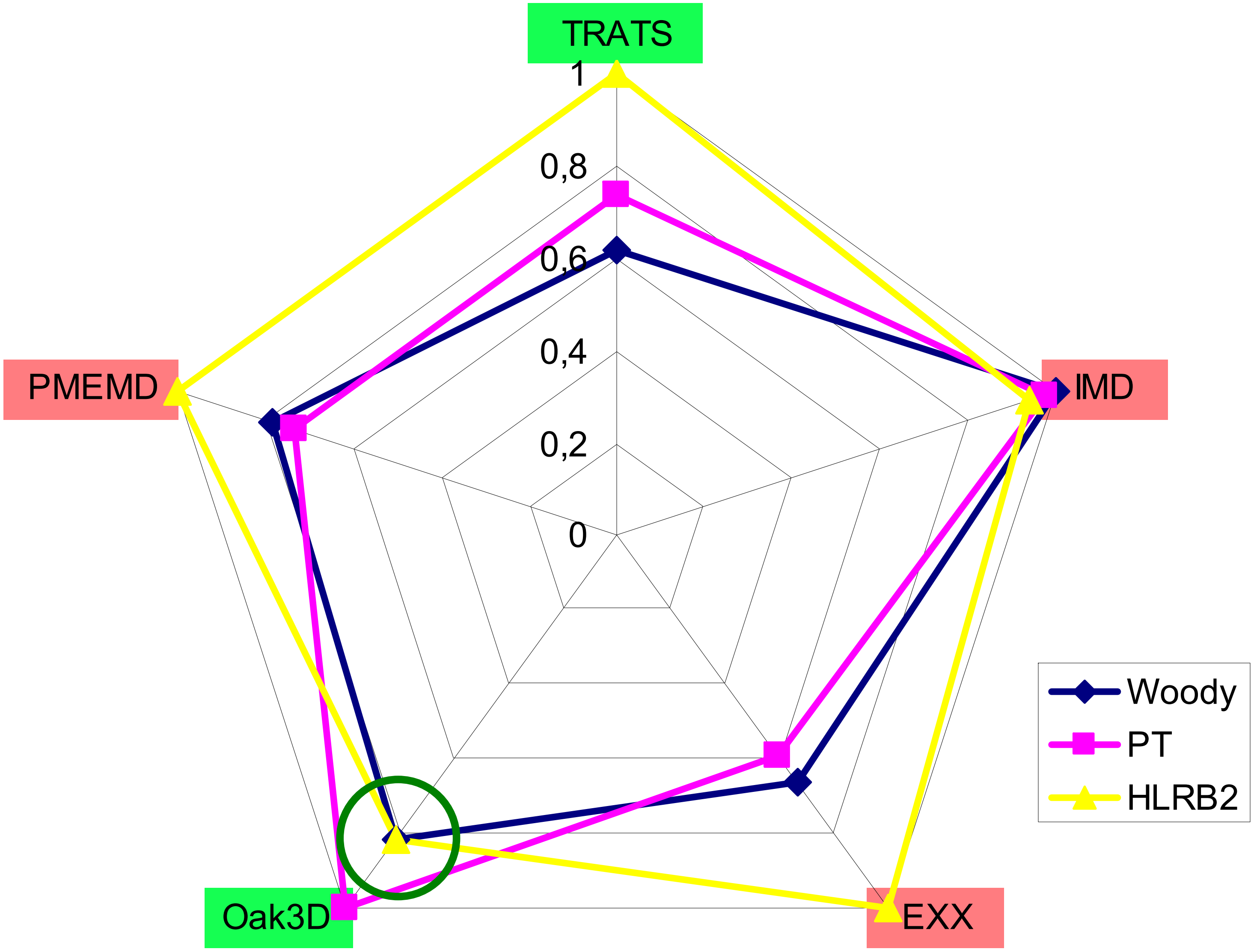
- Oak3D
  - FAU, theoretical physics
  - Time dependent HF for simulation of super-heavy nuclei
  - Dominated by small FFTs and some dense matrix-vector operations

- TRATS a.k.a. BEST
  - FAU, LSTM and HLRS
  - Lattice Boltzmann production code, highly scalable
  - Memory bound on standard microprocessors, compute bound on NEC SX (code balance ≈ 0.4 Words/Flop)
  - “Mother” of most RRZE LB benchmarks

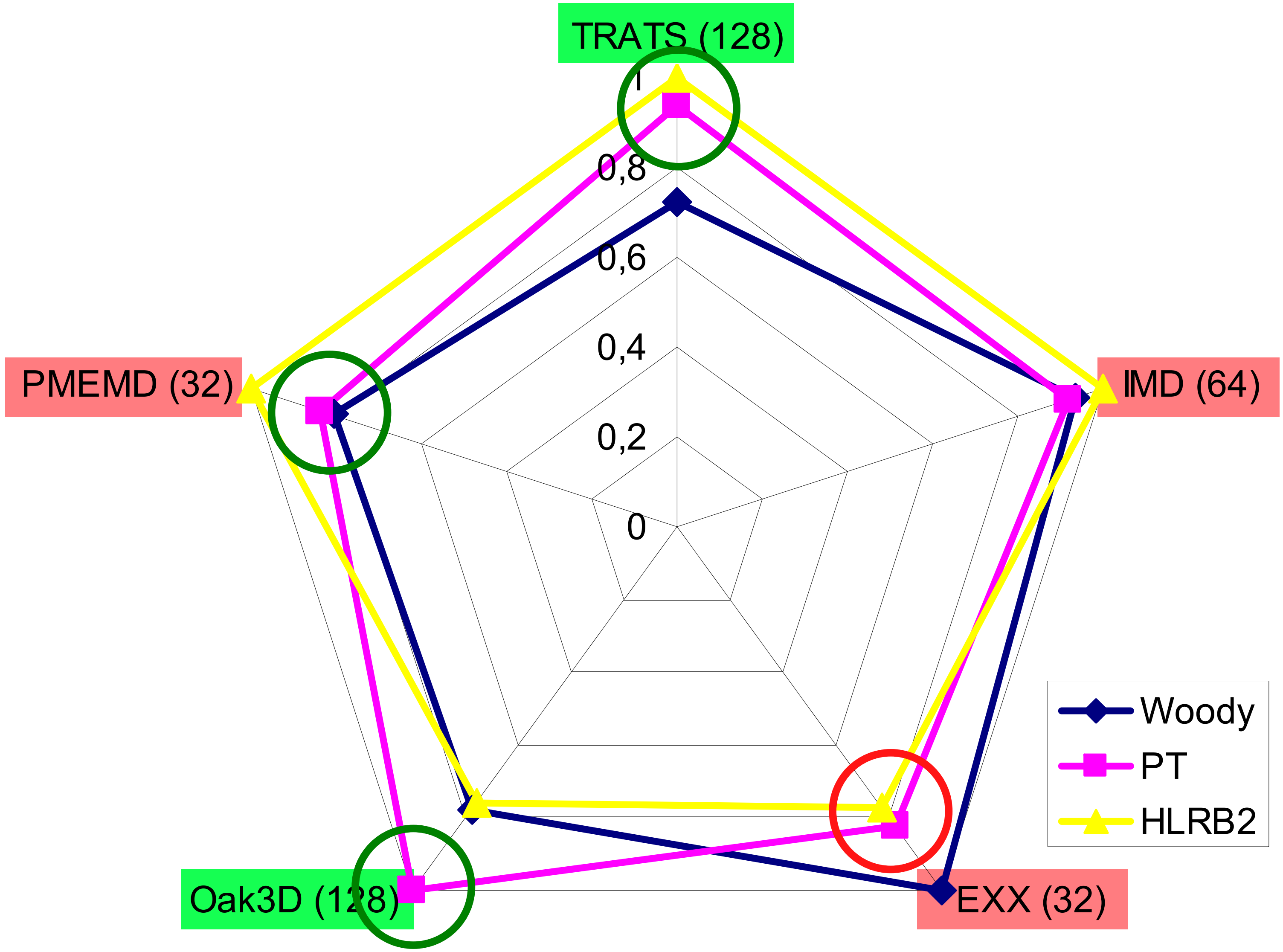
RZBENCH 1-core comparison



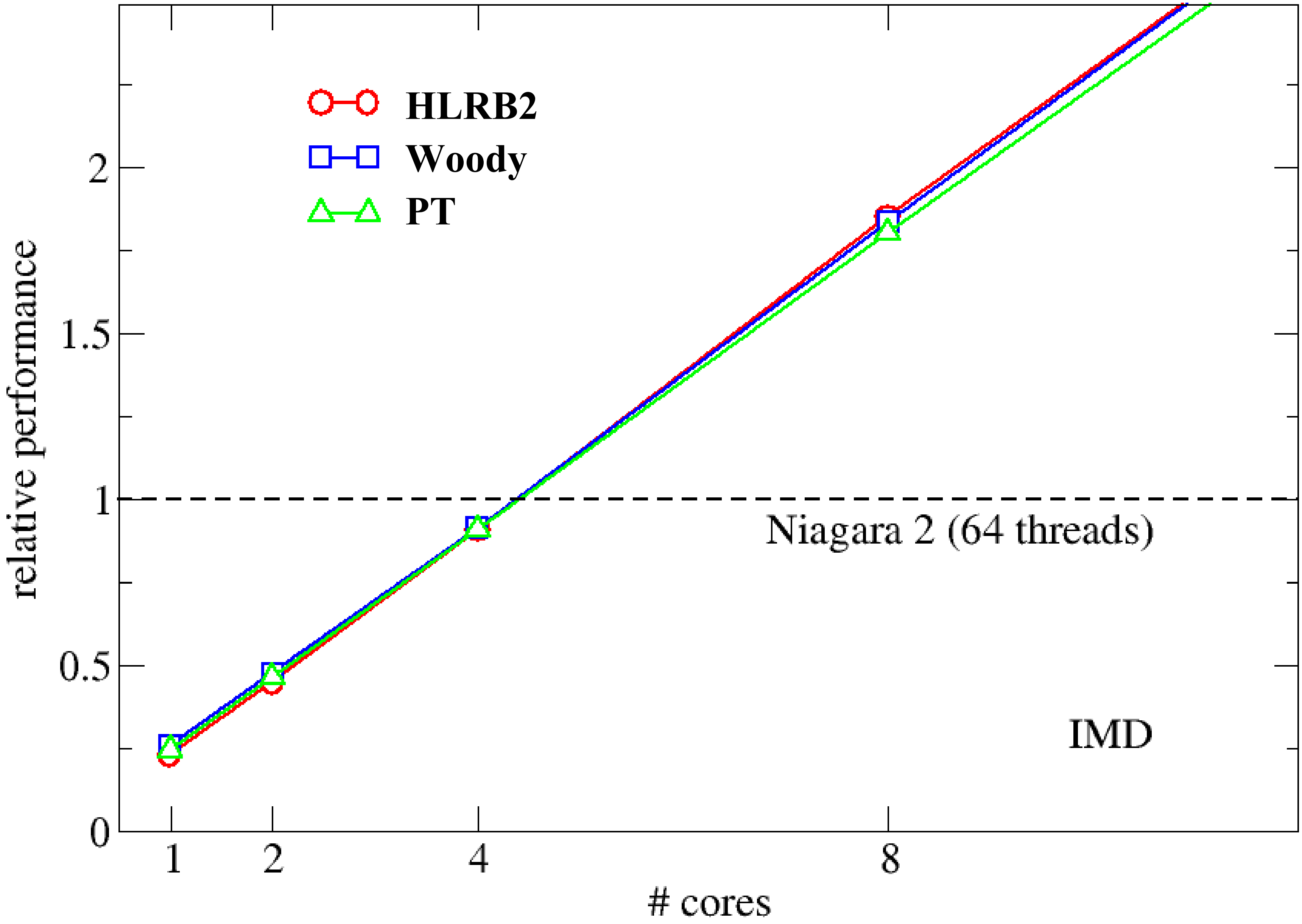
RZBENCH 1-socket comparison



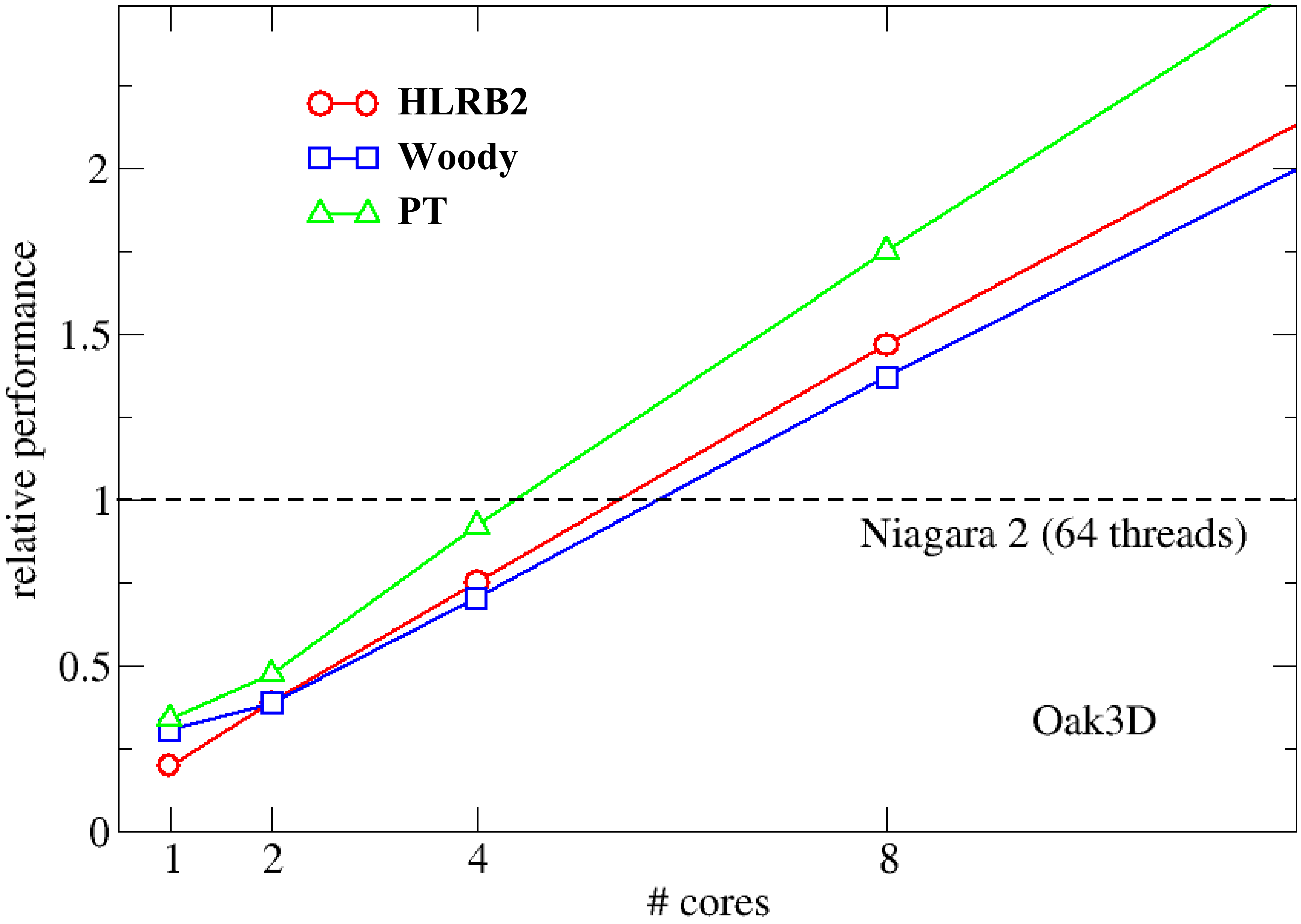
RZBENCH parallel comparison



How well does Niagara2 do?
IMD



How well does Niagara2 do? Oak3D



Conclusions

- One-socket clusters
  - Are not really off the price/perrformance “sweet spot”
  - Have simpler node design, less locality problems
  - Higher price tag is compensated by better bandwidth
    - memory and network!

- Sun UltraSPARC T2
  - Interesting design, “BlueGene on a chip”?
  - Aliasing problems on memory controllers must be carefully avoided
  - Weak (size, speed) L2 cache
  - Fastest CPU today?