TECHNISCHE
UNIVERSITÄT
DRESDEN

Center for Information Services and High Performance Computing (ZIH)

Performance Analysis of Computer Systems

Workloads and Benchmarks (continued)

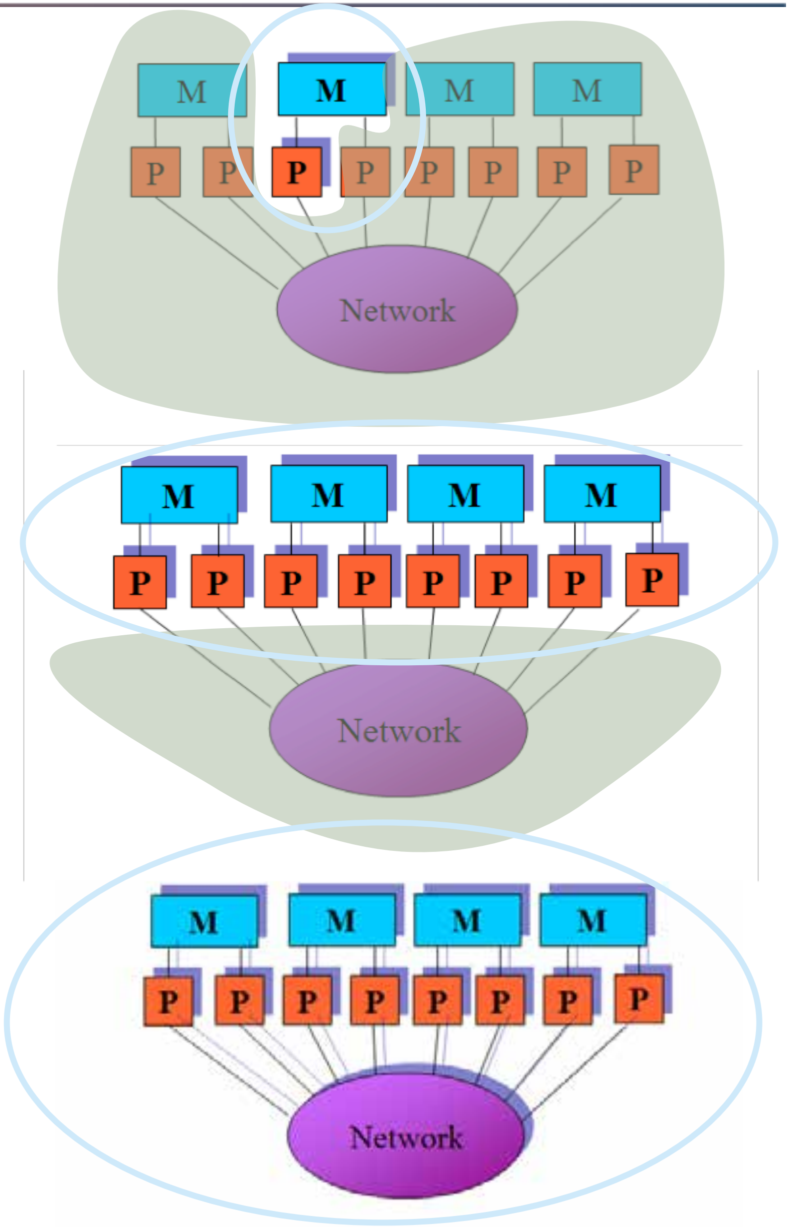
Holger Brunst (holger.brunst@tu-dresden.de)

Bert Wesarg (bert.wesarg@tu-dresden.de)

ZIH
Center for Information Services &
High Performance Computing

Tests on Single Processor and System

- **Local**
  - only a single processor performs computations
- **Embarrassingly Parallel**
  - all processors perform computations
  - NO explicit communication
- **Global**
  - all processors perform computations
  - explicit communication with each other



Outline

- Benchmarks
  - Main memory (Stream)
  - Floating point units (LINPACK, HPL)
  - File system (IOzone)
  - System interconnect (IMB)
  - HPC Challenge

Stream Benchmark

- Author: John McCalpin (“Dr Bandwidth”)
- John McCalpin “Memory Bandwidth and Machine Balance in High Performance Computers”, IEEE TCCA Newsletter, December 1995
- http://www.cs.virginia.edu/stream/
- STREAM: measure memory bandwidth with the operations:
  - Copy: a(i) = b(i)
  - Scale: a(i)=s*b(i)
  - Add: a(i)=b(i)+c(i)
  - Triad: a(i)=b(i)+s*c(i)
- STREAM2: measures memory hierarchy bandwidth with the operations:
  - Fill: a(i)=0
  - Copy: a(i)=b(i)
  - Daxpy: a(i) = a(i) +q*b(i)
  - Sum: sum += a(i)

Stream 2 properties

Kernel	Code	Bytes/ iter read	Bytes/ iter written	FLOPS/ iter
Fill	a(i)=q	0	8	0
Copy	a(i) = b(i)	8	8	0
DaXpY	a(i)=a(i) + q*b(i)	16	8	2
Sum	s = s + a(i)	8	0	1

Stream Results: TOP 10 in 2013

3.218 GB/s per core

```
STREAM Memory Bandwidth --- John D. McCalpin, mccalpin@cs.virginia.edu
Revised to Tue, Sep 17, 2013  5:28:07 PM

All results are in MB/s --- 1 MB=10^6 B, *not* 2^20 B

------------------------------------------------------------------------------------------
Sub. Date   Machine ID                          ncpus     COPY       SCALE         ADD        TRIAD
------------------------------------------------------------------------------------------
2012.08.14  SGI_Altix_UV_2000                   2048  6591669.0  6592082.0  7128484.0  7139690.0
2011.04.05  SGI_Altix_UV_1000                   2048  5321074.0  5346667.0  5823380.0  5859367.0
2006.07.10  SGI_Altix_4700                      1024  3661963.0  3677482.0  4385585.0  4350166.0
2013.03.26  Fujitsu_SPARC_M10-4S                1024  3474998.0  3500800.0  3956102.0  4002703.0
2011.06.06  ScaleMP_Xeon_X6560_64B               768  1493963.0  2112630.0  2252598.0  2259709.0
2004.12.22  SGI_Altix_3700_Bx2                   512   906388.0   870211.0  1055179.0  1119913.0
2003.11.13  SGI_Altix_3000                       512   854062.0   854338.0  1008594.0  1007828.0
2003.10.02  NEC_SX-7                              32   876174.7   865144.1   869179.2   872259.1
2008.04.07  IBM_Power_595                         64   679207.2   624707.8   777334.8   805804.6
2013.09.12  Oracle_SPARC_T5-8                    128   604648.0   611264.0   622572.0   642884.0
```

Stream Results: TOP 10 in 2006

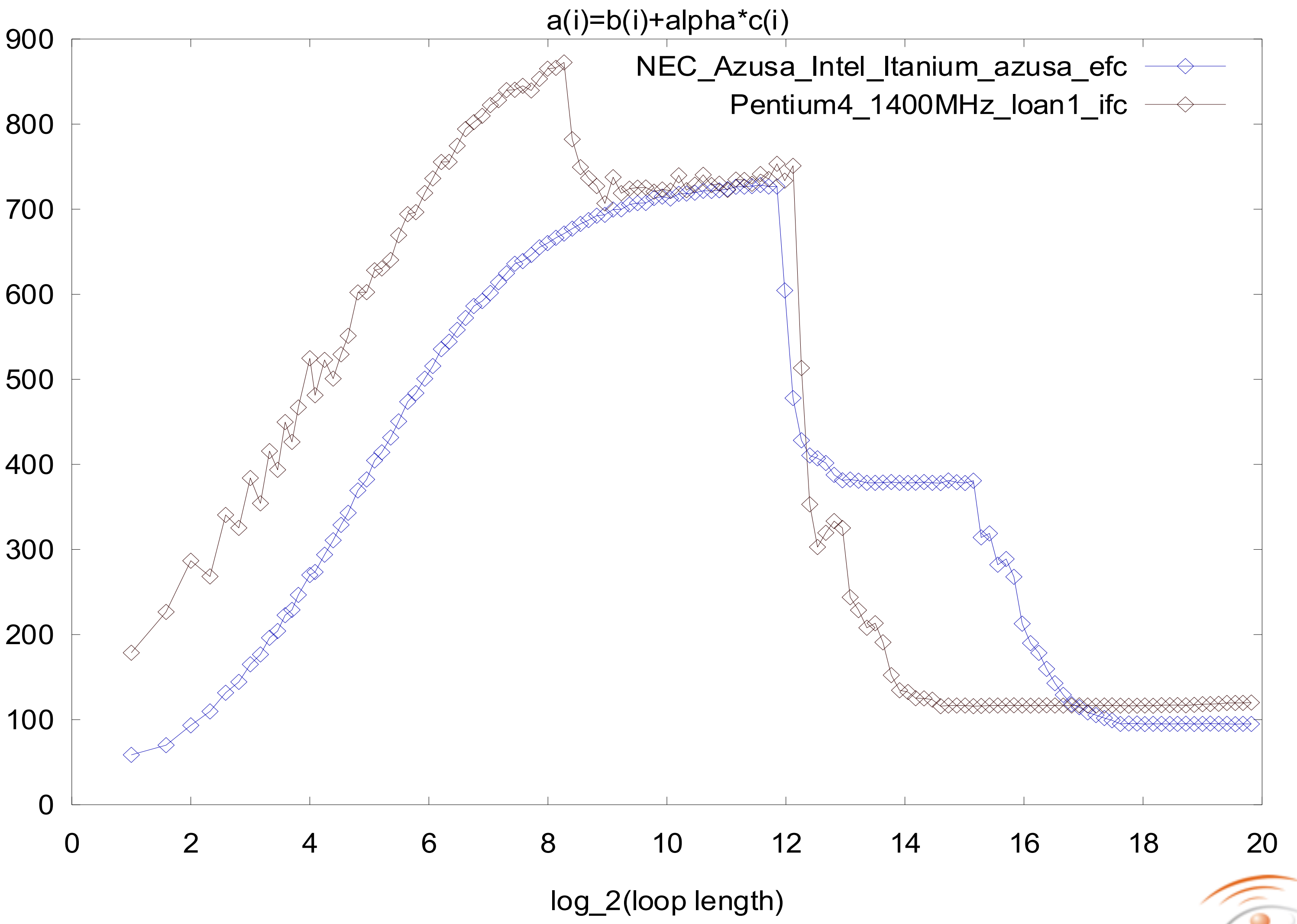
STREAM Memory Bandwidth --- John D. McCalpin, mccalpin@cs.virginia.edu
Revised to Tue Jul 25 10:10:14 CST 2006

All results are in MB/s --- 1 MB=10^6 B, *not* 2^20 B.

3.576 GB/s per core

```
----------------------------------------------------------------
Machine ID                        ncpus     COPY       SCALE        ADD       TRIAD
----------------------------------------------------------------
SGI_Altix_4700                     1024   3661963.0  3677482.0  4385585.0  4350166.0
SGI_Altix_3000                      512    906388.0   870211.0  1055179.0  1119913.0
NEC_SX-7                             32    876174.7   865144.1   869179.2   872259.1
NEC_SX-5-16A                         16    607492.0   590390.0   607412.0   583069.0
NEC_SX-4                             32    434784.0   432886.0   437358.0   436954.0
HP_AlphaServer_GS1280-1300           64    407351.0   400142.0   437010.0   431450.0
Cray_T932_321024-3E                  32    310721.0   302182.0   359841.0   359270.0
NEC_SX-6                              8    202627.2   192306.2   190231.3   213024.3
IBM_System_p5_595                    64    186137.0   179639.0   200410.0   206243.0
HP_Integrity_SuperDome              128    154504.0   152999.0   169468.0   170833.0
```

Stream 2 Results



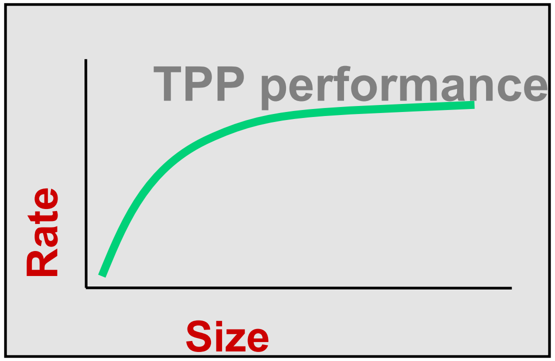
Outline

- Benchmarks
  - Main memory (Stream)
  - Floating point units (LINPACK, HPL)
  - File system (IOzone)
  - System interconnect (IMB)
  - HPC Challenge

LINPACK

- Originally
  - Library for numerical linear algebra
  - 1970 – 1980
  - Jack Dongarra *et al.*
  - Fortran
  - Included performance test program
  - Successor: LAPACK
- Today
  - Since 1993 benchmark for Supercomputers (TOP500)
  - FLOPS: Number of **fl**oating point **op**erations per **s**econd

HPL Benchmark



- **H**igh **P**erformance **L**inpack
- Third Linpack test
- Algorithmic
- Solve Ax=b, random dense linear system
  - Uses LU decomposition with partial pivoting
  - Based on the ScaLAPACK routines but optimized
  - Scalable in the sense that the parallel efficiency is maintained constant with respect to the per processor memory usage (weak scaling)
  - In double precision (64-bit) arithmetic
  - Run on all processors
  - Problem size (N) set by user/vendor

HPL Metrics in TOP500

- N_{max} – the size of the chosen problem run on a machine
- R_{max} – the performance in Gflop/s for the chosen size problem
- $N_{1/2}$ – the size where half the R_{max} execution rate is achieved
- R_{peak} – the theoretical peak performance Gflop/s for the machine

HPL Background

- Requirements
  - MPI (Message Passing Interface)
  - BLAS (Basic Linear Algebra Subprograms)
- Resources
  - http://www.netlib.org/benchmark/hpl/
  - http://www.netlib.org/utk/people/JackDongarra/faq-linpack.html

Outline

- Benchmarks
  - Main memory (Stream)
  - Floating point units (LINPACK, HPL)
  - File system (IOzone)
  - System interconnect (IMB)
  - HPC Challenge

Linux & File System I/O

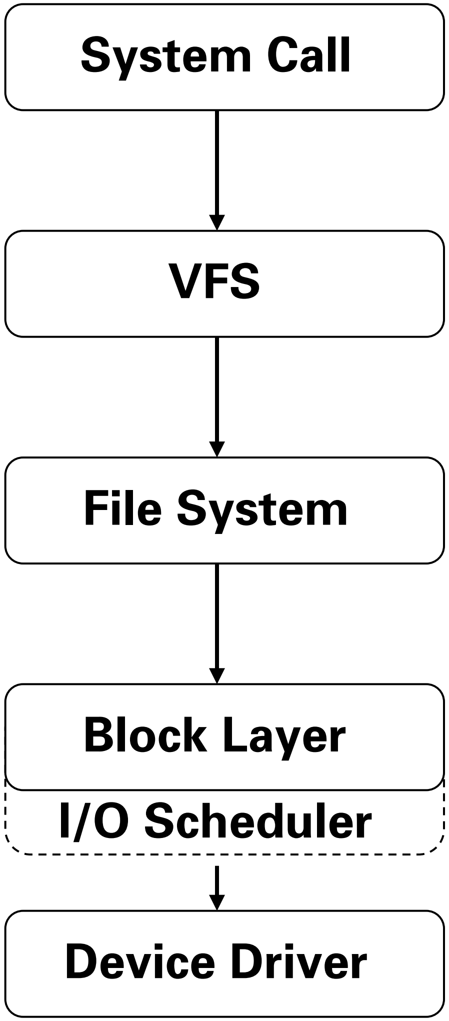
read()

vfs_read()

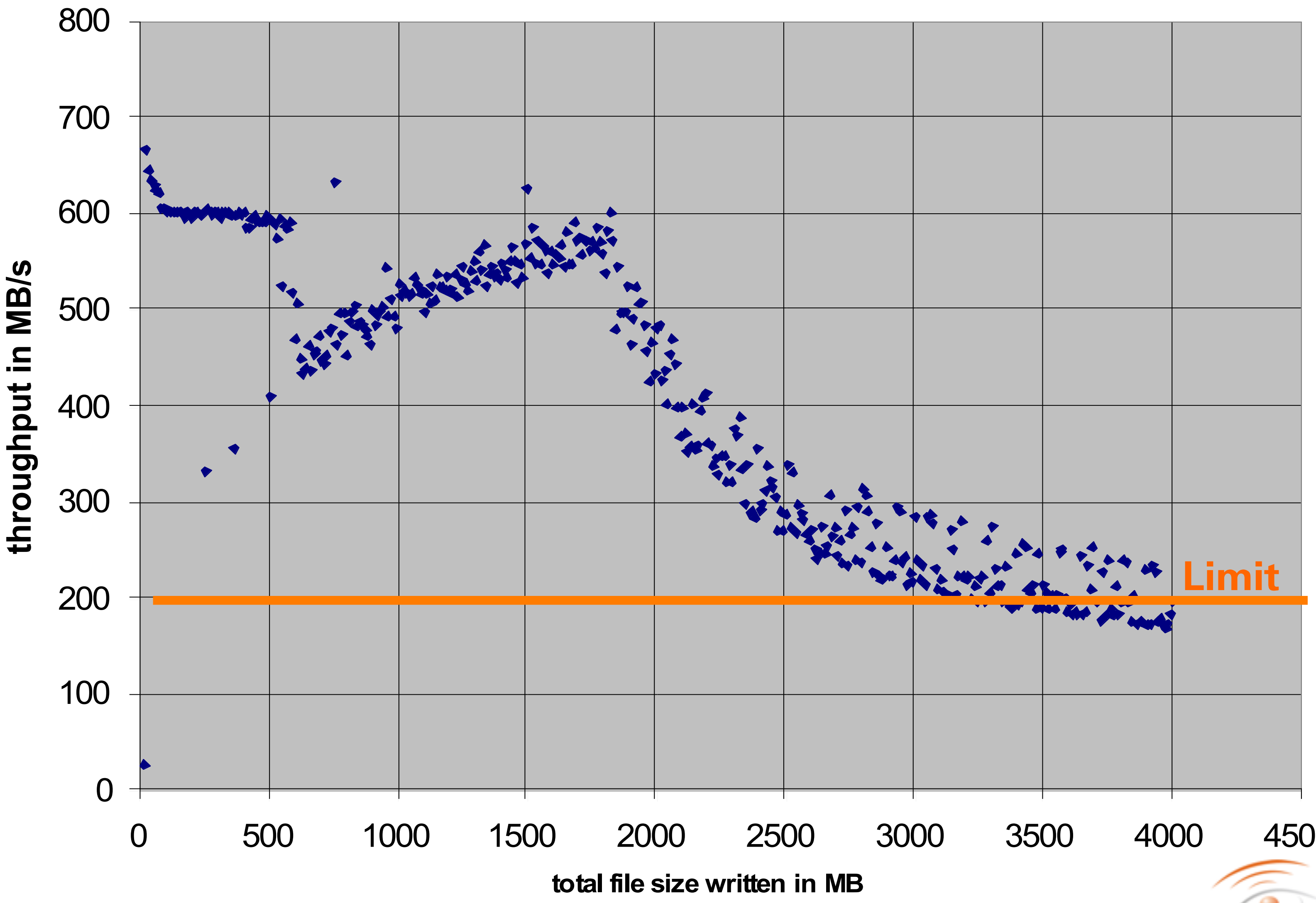
file->f_op->read()

submit_bio()

q->reuest_fn()



Influence of Cache Buffers



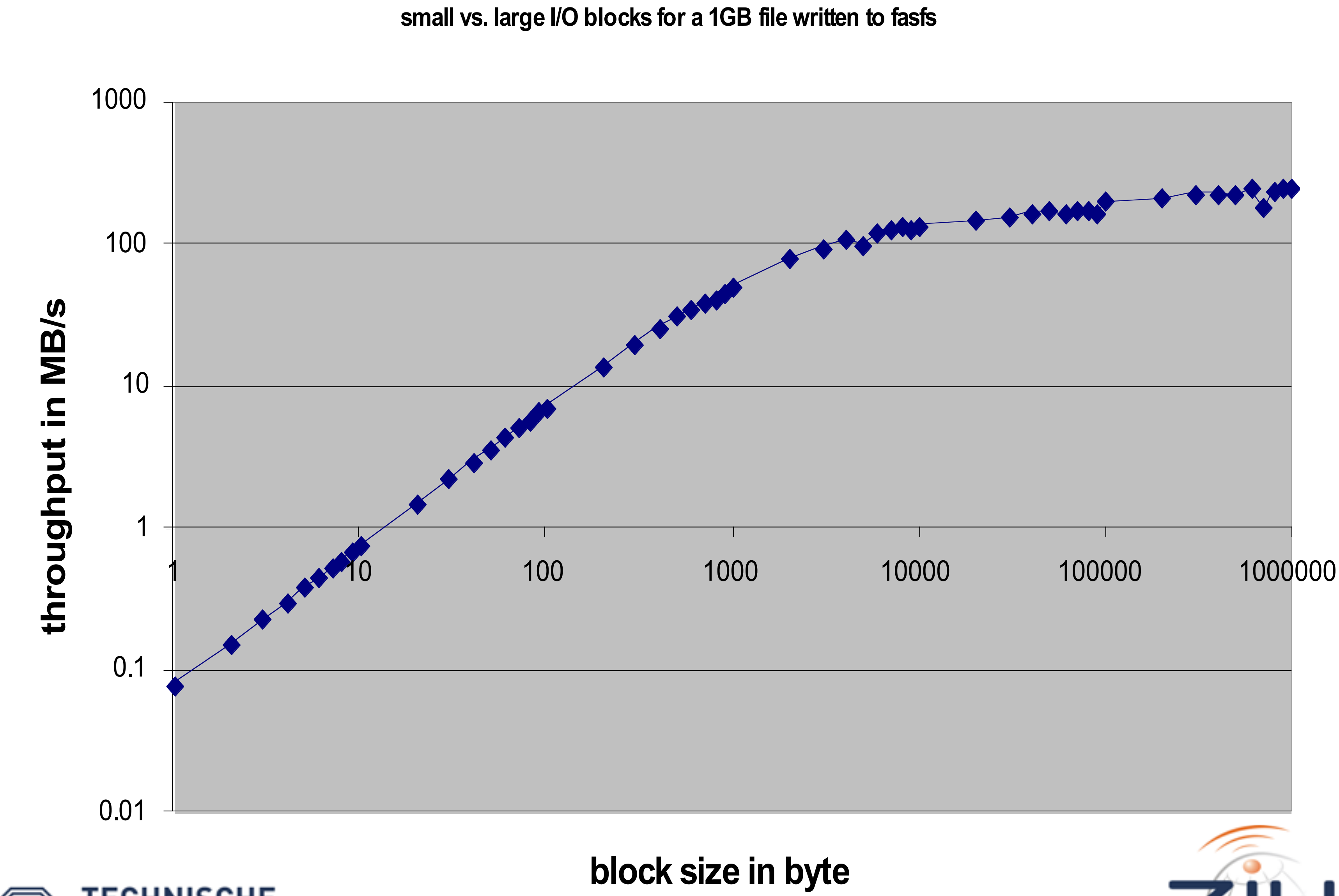
Characterization by Application

- Raw (disk) access
- Unbuffered
  - open (IO_DIRECT), close
  - read/write
- Buffered
  - fopen, fclose
  - fread, fwrite
- Standard Libraries
  - MPI-I/O
  - (parallel) NetCDF (network common data form)
  - HDF5 (Hierarchical Data Format Version 5)

Other I/O Characterizations

- Operations per second of
  - open, close
  - stat (dir)
  - unlink (delete)
- Size
  - Small vs. large data volumes
- Access pattern
  - Read, write
  - Repeated read/write
- Target
  - SSDs, Disks, Tapes

Small I/O vs. Large I/O



Measuring Physical I/O Bandwidth

- Linux uses free main memory (max. 8 GB per node) for I/O caching
- Example:
  - Physical I/O bandwidth: approx. **0.1 GB/s**
  - Bandwidth to I/O cache is **10 GB/s**
  - One process writes big data blocks
- Measure bandwidth in GB/s
  - *size of data / duration*

- Question: How many data need to be written to measure the physical bandwidth with a deviation < 10% ?

Typical Benchmark Parameters

- Hardware
- Operating system
- File system
- Buffered, unbuffered
- Type of operation
- Access pattern
- Usage of caches
- Parallelism
- Repetition

Typical Goals

- Evaluation of metadata rate
  - How many files can be opened per second in the same directory by N clients?
  - How many files can be opened per second in N directories by N clients
  - Likewise, but with directories

Typical Goals Contd.

- Evaluation of CPU overhead
  - Reading/writing of small buffers from/to file system
  - Reading/writing with different access patterns

- Evaluation of maximum bandwidth
  - Reading/writing of big blocks with
    - one task
    - multiple tasks
  - Optional 1:1 mapping of tasks and #cores or #interfaces

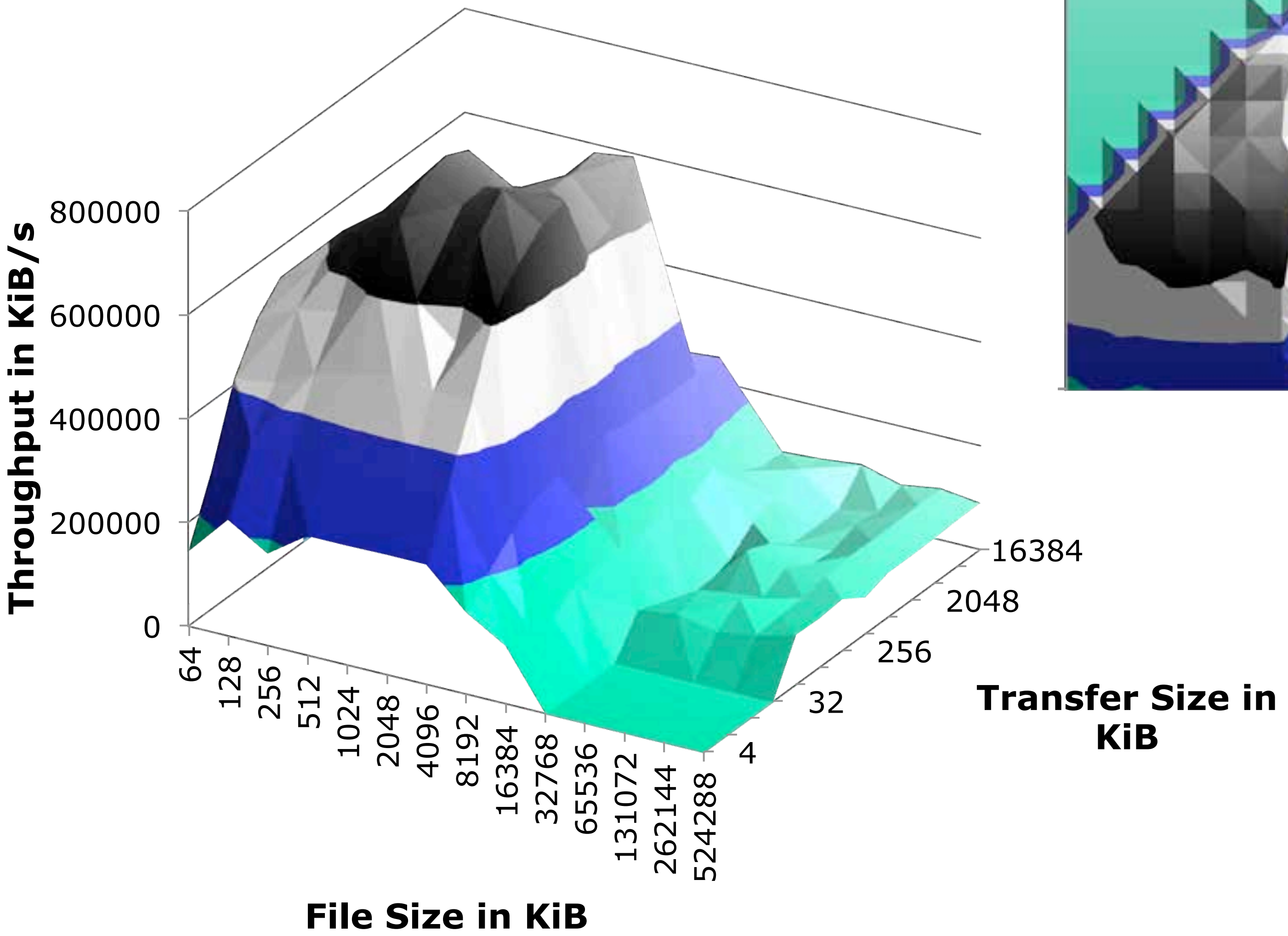
IOzone Benchmark

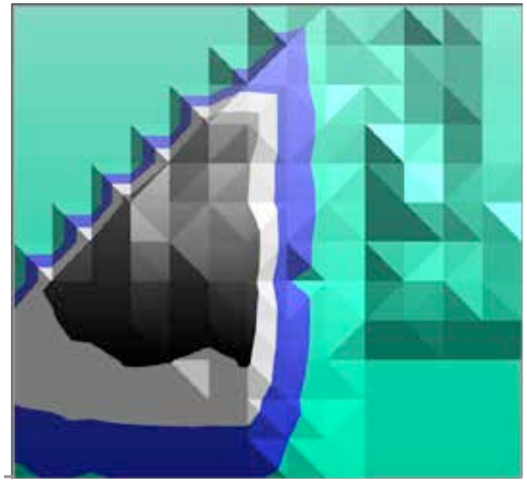
- File system benchmark
- Written in C
- Metric: Throughput in bytes/s and Latency
  - Latency maps (duration + offset)
- Workload: Read/write a file of a given size
- Resources
  - http://www.iozone.org/

IOzone Benchmark

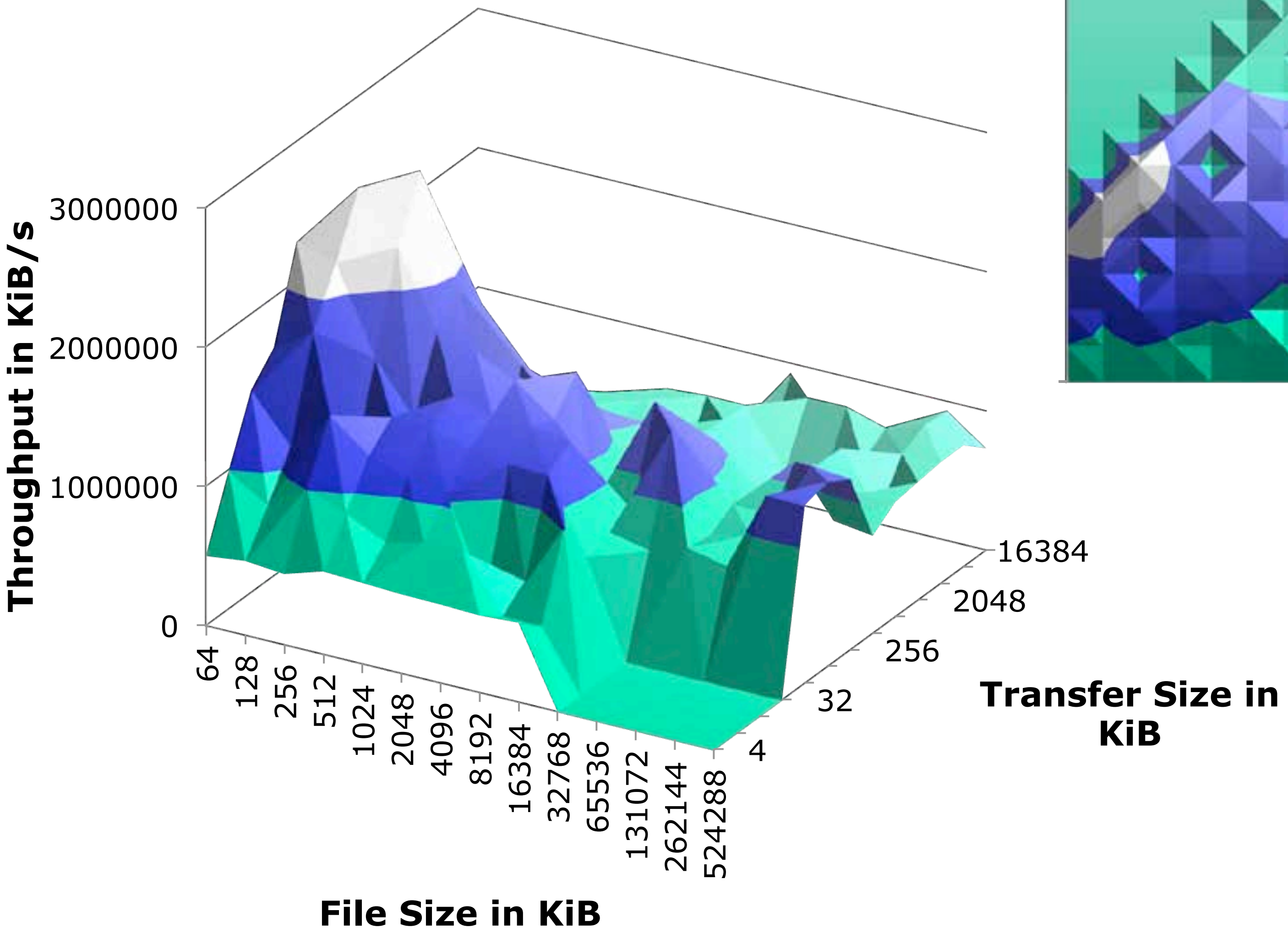
- Parameters/Factors
  - File size
  - Transfer size (block size)
- Tests
  - read/write
  - re-read/re-write
  - fread/fwrite
  - pread, mmap
  - aio_read/write
  - strided or random read

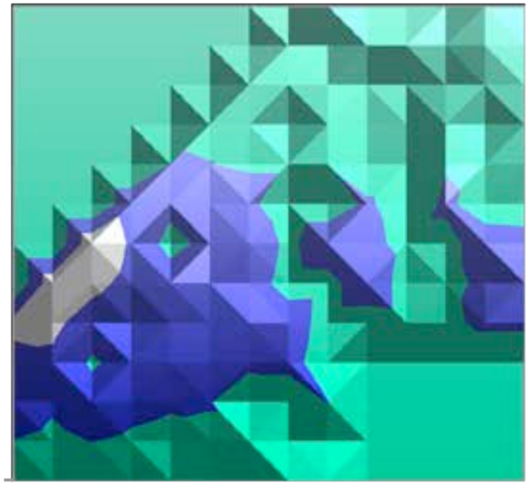
IOzone Write Benchmark Results (Mac G5)



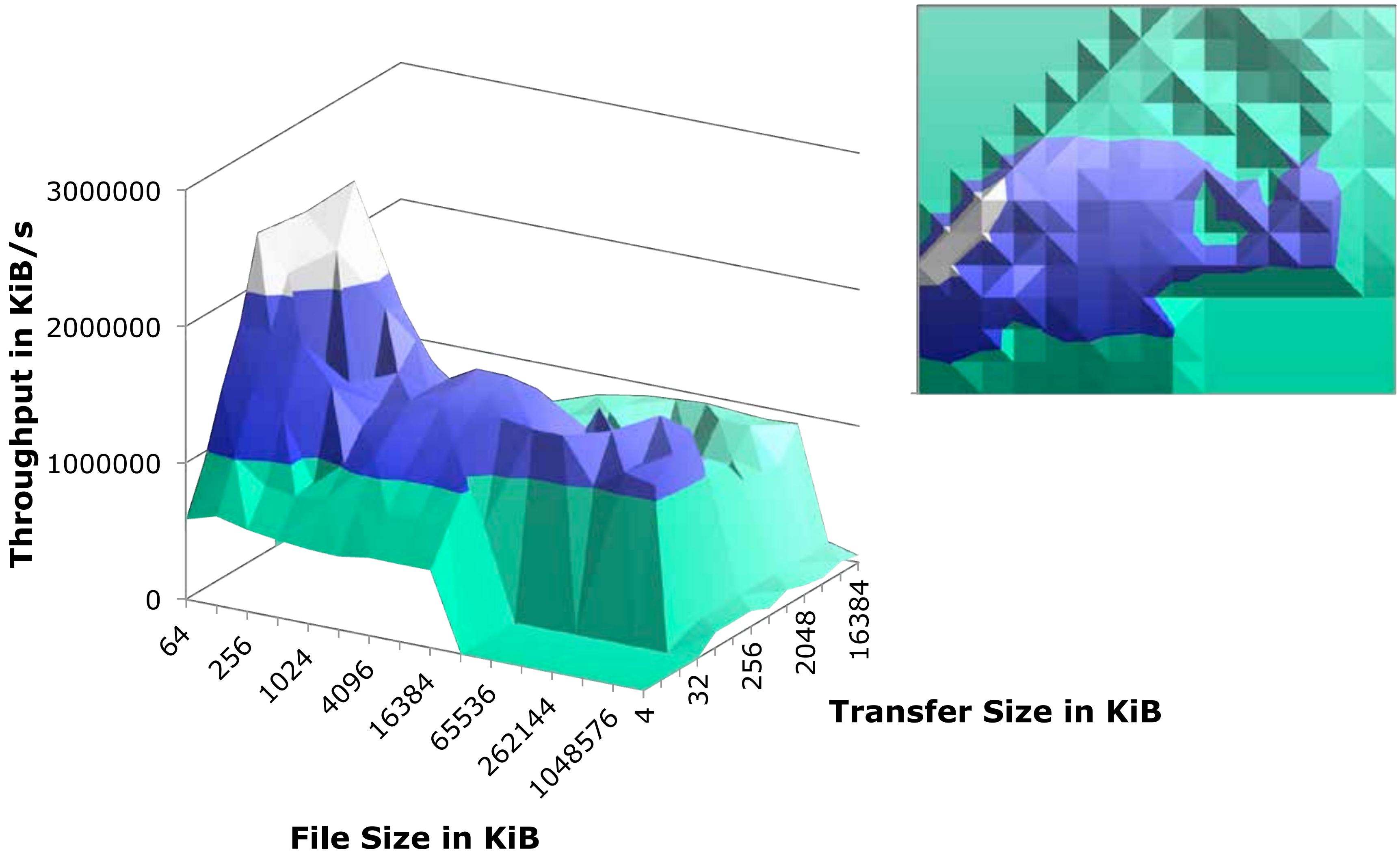


IOzone Read Benchmark Results (Mac G5)





Read Results with File Size = 2GiB



Outline

- Benchmarks
  - Main memory (Stream)
  - Floating point units (LINPACK, HPL)
  - File system (IOzone)
  - System interconnect (IMB)
  - HPC Challenge

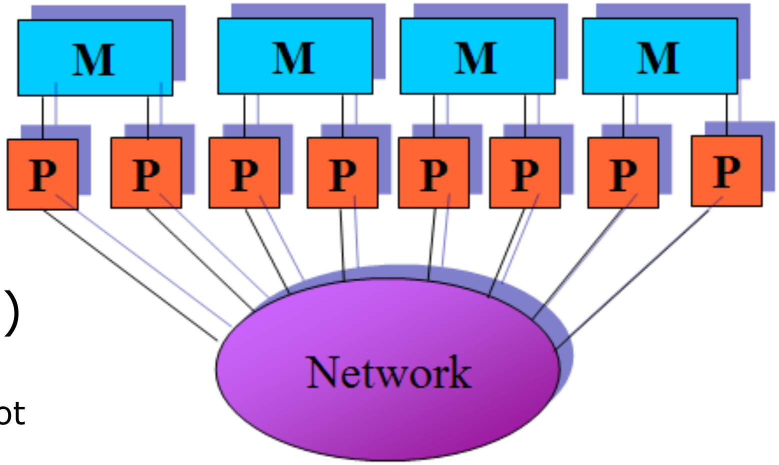
Intel MPI Benchmark

- Evaluation of MPI implementations

- MPI = **M**essage **P**assing **I**nterface (Standard)
  - Explicit communication between processes
- Implementations
  - MPICH
  - OpenMPI
  - Intel MPI
  - Microsoft MPI
- Side remark: Former *Pallas MPI Benchmark*

Intel MPI Benchmark

- Metrics
  - Throughput
  - Time
- Categories
  - Single Transfer (S_)
    - Source A to target B
  - Parallel Transfer (P_)
    - N sources and targets (conc.)
    - $T_{tot}=max(t_i)$, $B_{tot}=sum(p_i)/T_{tot}$
  - Collective Transfer (C_)
    - Test collectives as in MPI
    - No throughput! **Why not?**



Outline

- Benchmarks
  - Main memory (Stream)
  - Floating point units (LINPACK, HPL)
  - File system (IOzone)
  - System interconnect (IMB)
  - HPC Challenge

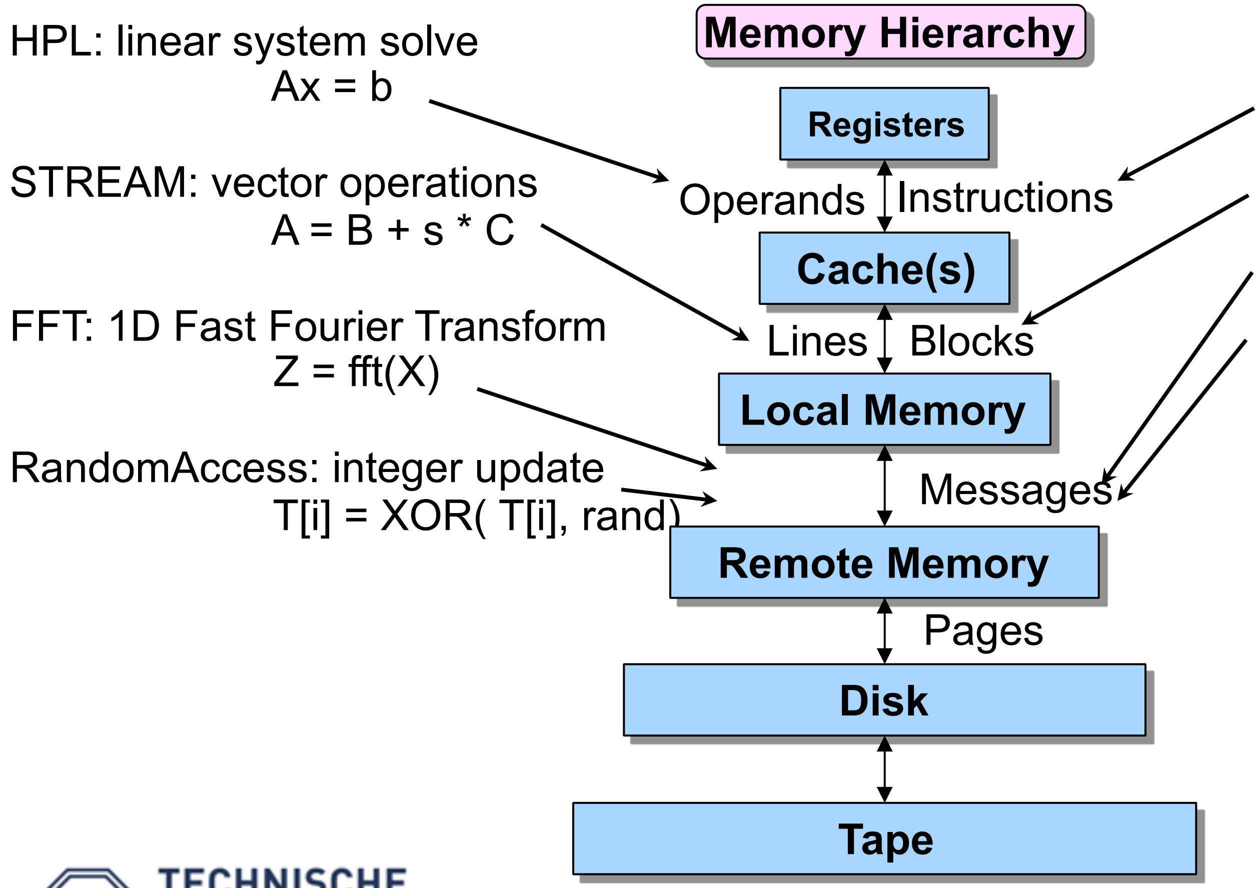
HPC Challenge

Basically 7 benchmarks:

1. HPL (LINPACK) — MPI Global ($Ax = b$)
2. Matrix Multiply — single CPU and Embarrassingly parallel (EP)
3. STREAM — Local, single CPU
   *STREAM — EP
4. PTRANS ($A \quad A + B^T$) — MPI Global
5. RandomAccess — Local, single CPU
   *RandomAccess — EP
   RandomAccess — MPI Global
6. BW and Latency (based on b_efd) — MPI Global
7. FFT — single CPU, EP, and MPI Global

Performance Targets

- HPCC was developed by HPCS
- Each benchmark focuses on a different part of the memory hierarchy



HPL: linear system solve
Ax = b

STREAM: vector operations
A = B + s * C

FFT: 1D Fast Fourier Transform
Z = fft(X)

RandomAccess: integer update
T[i] = XOR(T[i], rand)

Max	Relative
2 Pflop/s	8x
6.5 Pbyte/s	40x
0.5 Pflop/s	200x
64000 GUPS	2000x