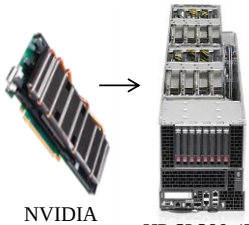
GPU Acceleration of Matrix Algebra

Dr. Ronald C. Young
Multipath Corporation

fmslib.com

FMS Performance History

DEC VAX 11/780



NVIDIA M2070 HP SL390 4U

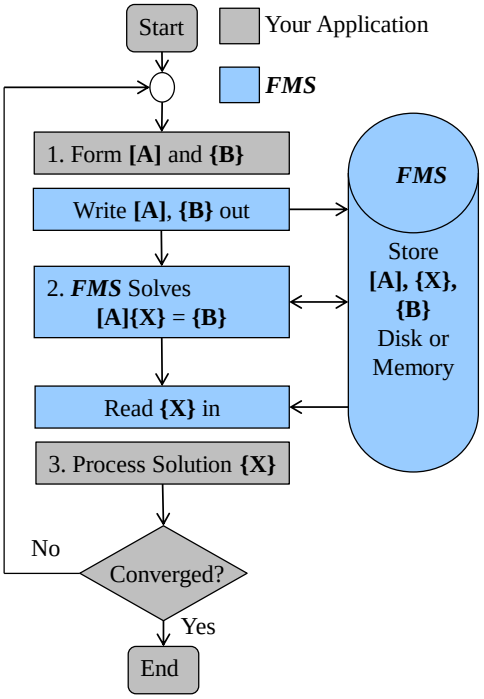
Machine	Year	Flops
DEC VAX	1978	97,000
FPS 164	1982	11,000,000
FPS 164-MAX	1985	341,000,000
X86 core	2011	10,000,000,000
NVIDIA GPUs	2011	2,200,000,000,000 (2.2 Tflops)

22 Million times faster than a VAX 11/780

fmslib.com

- ***FMS*** got its start in 1978 as a Floating Point Systems (FPS) matrix algebra package.
- Floating Point Systems made array processors which attached to VAX computers to provide improved floating point performance in hardware. At that time most commercial computers performed floating point operations is software (except for CRAY and CDC).
- ***FMS*** achieved near 100% efficiency by implementing a hardware multiply-accumulate instruction for dot products.
- In 1985 the first 64-bit adder and multiplier chips became available. Exploiting the broadcast interconnect properties of matrix algebra, Floating Point Systems produced the Matrix Algebra Accelerators which provided up to 30 additional multiply-accumulate pipelines for ***FMS*** applications.
- Today (2011) a typical core in an Intel processor performs at about 10 Gflops (2.5 GHz x 4 Flops/cycle). That will double with the AVX cores which perform 8 Flops/cycle.
- A SL390 having 8 NVIDIA Fermi GPU's and requiring ½ of a 4U cabinet achieves 2.2 Tflops. A cabinet with 20 nodes would achieve 44 Tflops. That will increase with the next generation GPU (Kepler).
- While new features have been added, the ***FMS*** application program interface has remained the same. Applications written for the VAX/FPS system will run on a Intel/GPU system with a recompile and linking to the current ***FMS*** libraries.

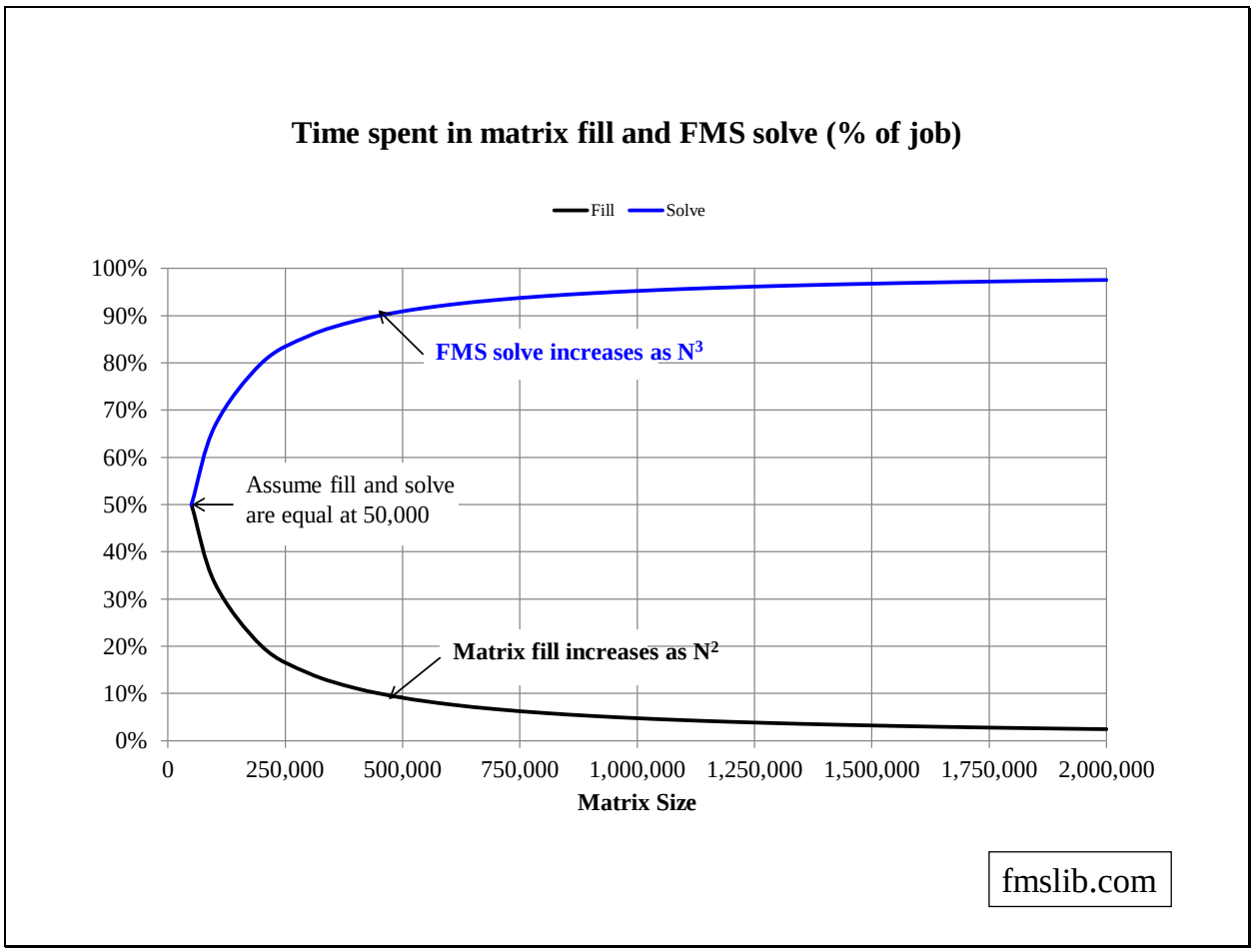
FMS Application Program Interface



- ***FMS*** captures data is it is generated.
- Specify data by rows, columns, blocks, finite elements or a call-back routine you provide.
- Methods to assemble, factor, solve, multiply.
- Methods to read results.
- Application part (grey) increases as N^2.
- ***FMS*** part (blue) increases as N^3.

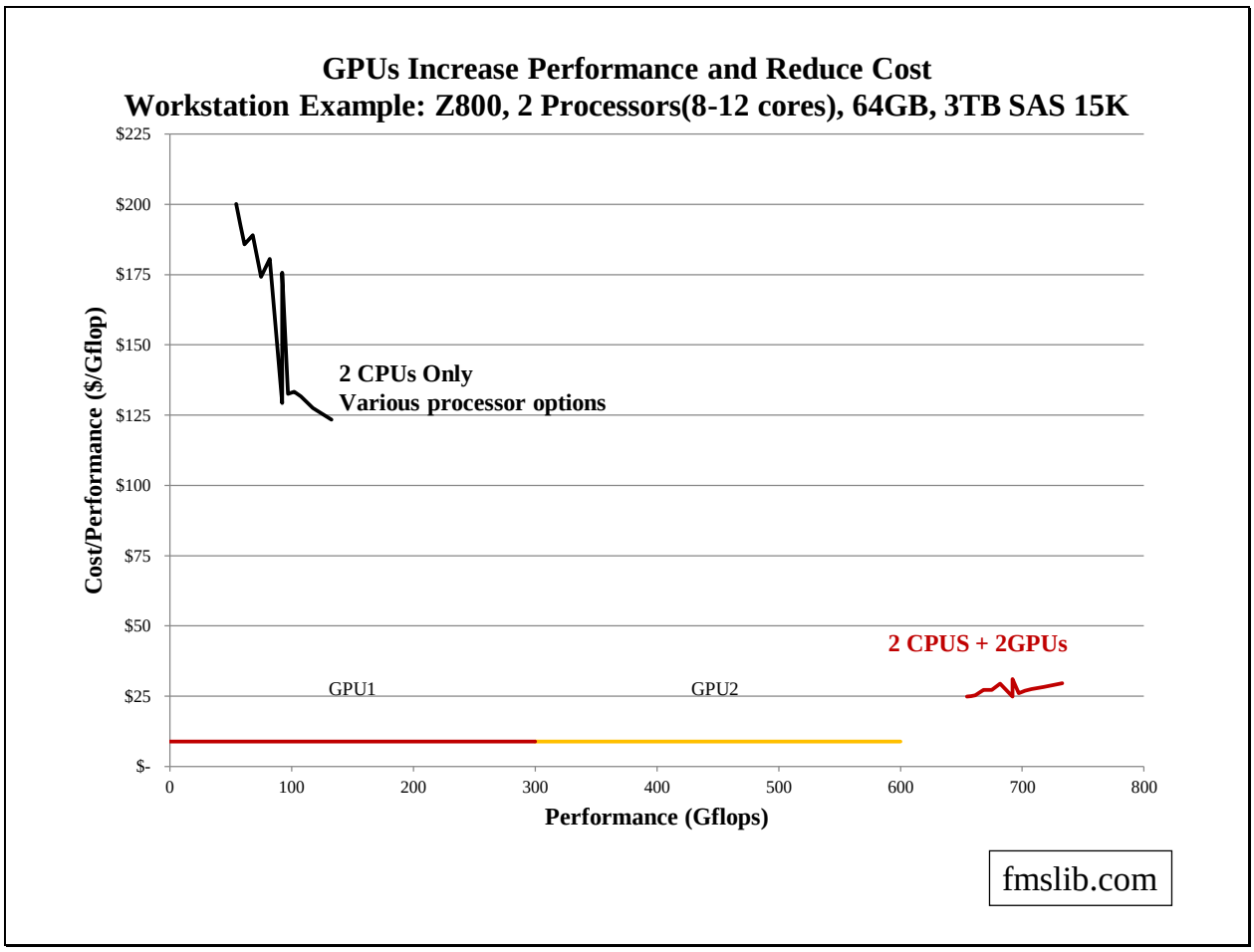
fmslib.com

- One of the main uses of ***FMS*** is for solving boundary value problems in production applications.
- These applications generally involve 3 steps:
  - Reading in the data and forming the matrix [A] and vector(s) {B}. ***FMS*** includes utilities for specifying the matrix data by rows, columns, blocks, finite elements or subroutines you provide. Note that if the problem exceeds memory when you solve it, it is also exceeds memory when you form it. During this stage ***FMS*** captures the data as it is being generated and stores it on disk *or memory* in a format that is most efficient for the target machine.
  - The second stage is to perform matrix algebra operations on the data. This step is performed by ***FMS***. Methods are included to assemble, factor, solve and multiply.
  - In the third stage, the application reads the data back from ***FMS***.
- If the problem is converged (the coefficients of [A] do not depend on the solution {X}) you are done. Otherwise a new [A] and {X} are formed and the process is repeated.
- The computations performed by ***FMS*** is step 2 involve a large number of regularly ordered floating point operations, which are ideally suited for a numerical coprocessor.
- For fully coupled problems, the time required for problem formation in Step 1 increases as the square of the problem size while the time required for the ***FMS*** solution in Step 2 increases as the cube of the problem size.

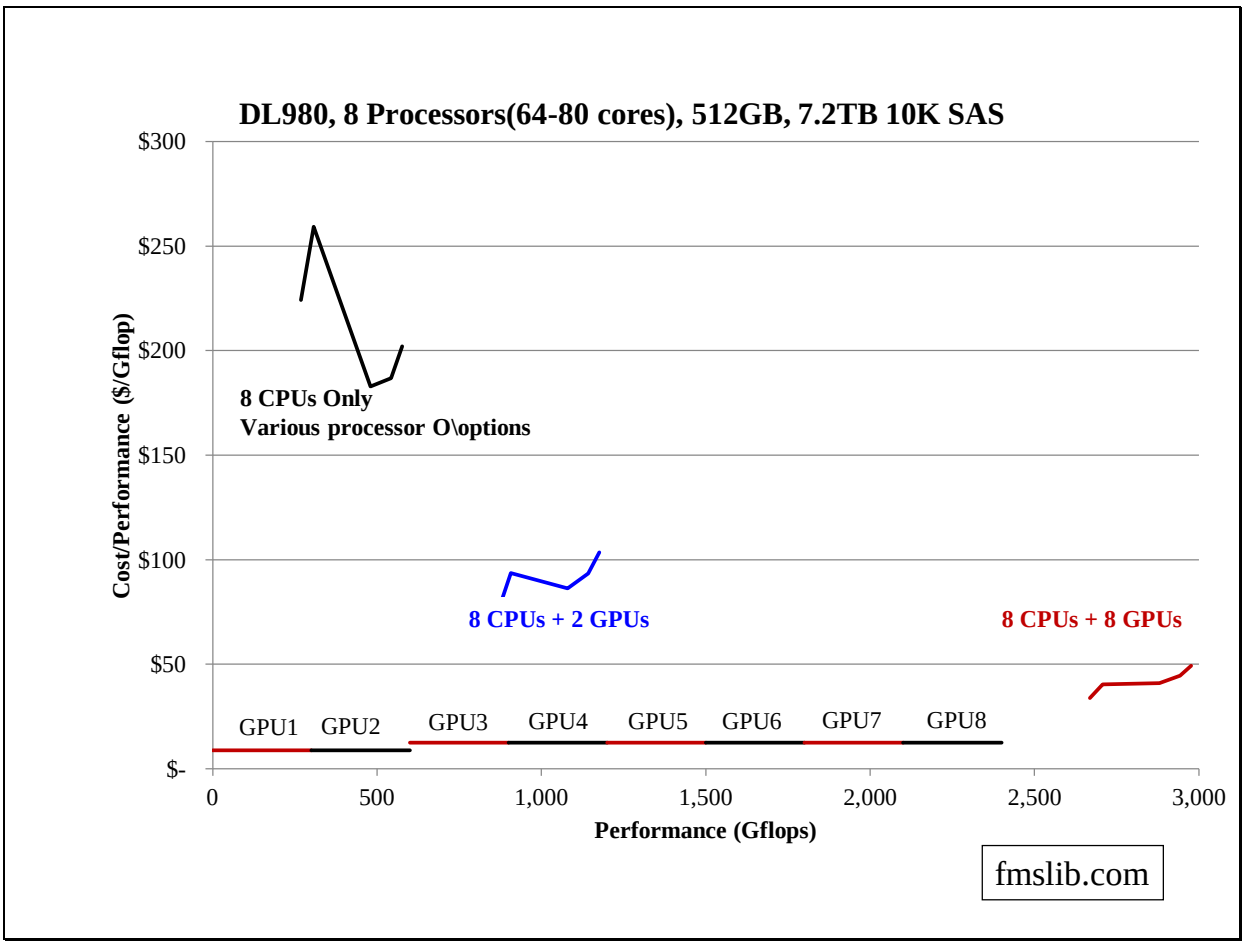


- As the problem size increases, the ***FMS*** part grows as the cube of the problem size while the rest of the application grows linearly or as the square.

- If the ***FMS*** and application parts are equal at 50,000 (meaning that it takes 50,000 cycles to generate each matrix element), the above curves show how the percentage of work changes with problem size.



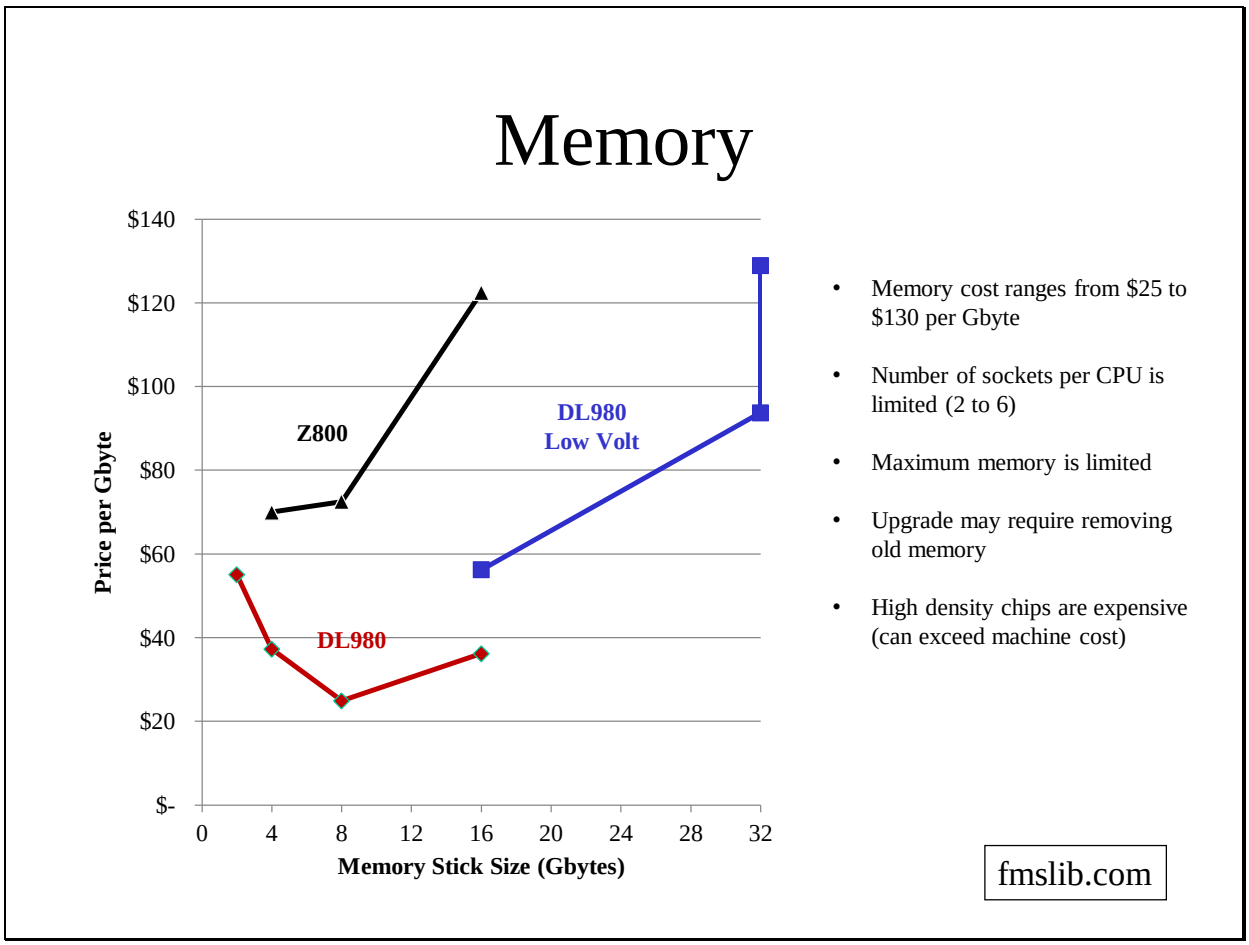
- As a base configuration, start with a HP Z800 workstation having (2) processors, 64 GBytes of memory and (5) 600 GByte SAS 15K disks.
- The system cost is about $200/Gflop of actual performance, based on HP's WEB site.
- Some benefit is achieved by processor upgrades, reducing the cost to $125/Gflop and extending performance to 130 Gflops
- Each NVIDIA C2050 provides in excess of 300 Gflops at $8.33/Gflop.
- GPU chips have a high percentage of transistors devoted to adders and multipliers.
- ***FMS*** operates the CPUs and GPUs in parallel, extending performance and reducing operational cost.



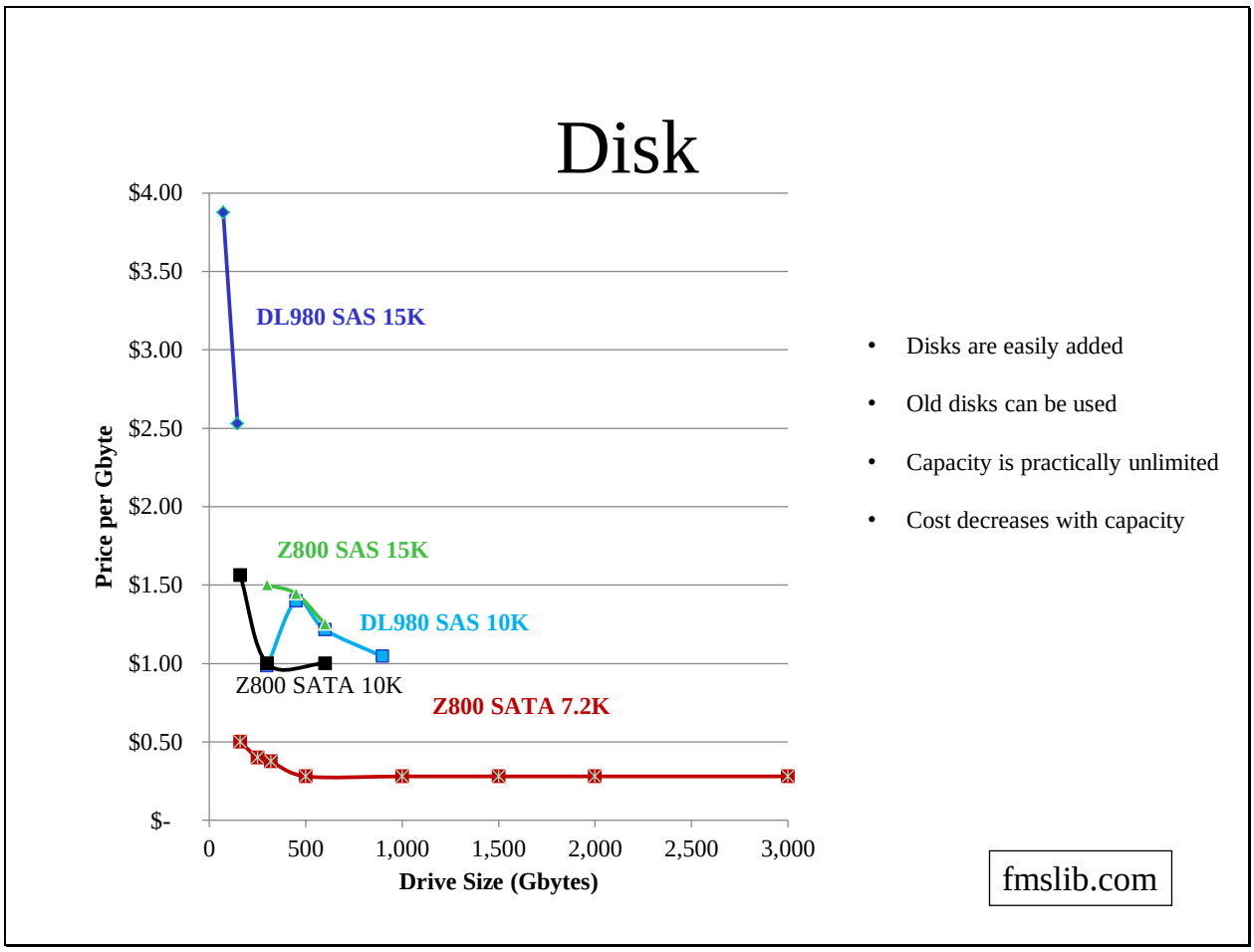
- As a base configuration start with a DL980 having 8 processors, 512GBytes of memory and (8) 900GByte 10K SAS disks.
- Depending on processor model, the cost/performance is between $180 and $225/Gflop.
- The maximum performance is 575 Gflops.
- (2) NVIDIA M2050 GPUs can be installed directly in the DL980, improving the cost/performance and extending the performance beyond 1 Tflop.
- (8) GPUs can be attached to the (2) PCI slots using expansion cabinets, extending the performance beyond 2.5 Tflops.

Data Storage Options

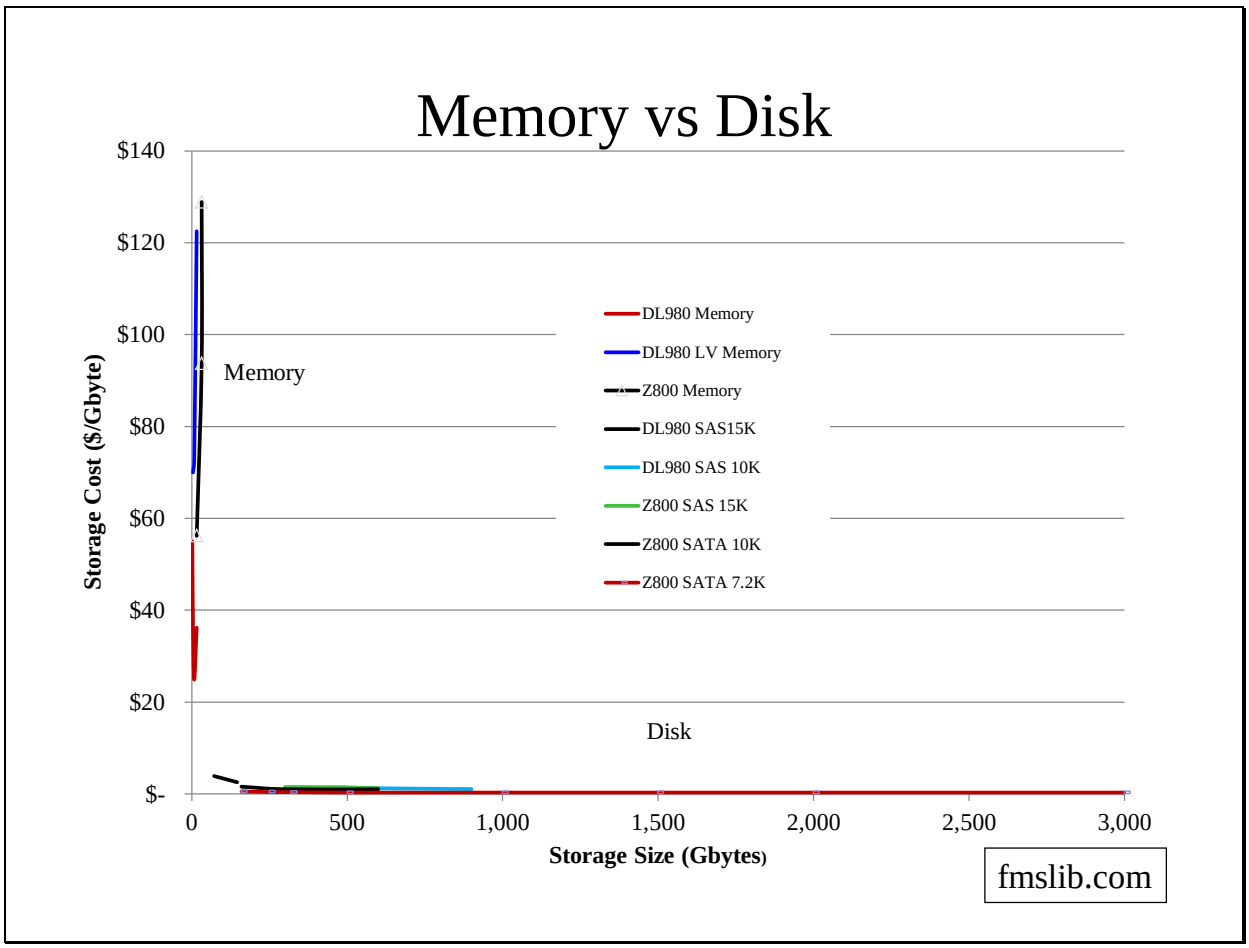
- GPUs provide the capability of solving larger problems.
- Memory may be insufficient or too expensive for storing the increased data.
- Disks are 100 times less expensive than memory and can be easily expanded.
- The reuse of data in matrix algebra can match high speed computation with slower disk transfer rates.



- The current cost for workstation or server memory is $25 to $130 per GByte, based on HP's WEB site.
- Memory based on denser chip technology offers more capacity, but at a higher price per unit of storage.
- The maximum amount of memory which can be installed in a system is limited by the number of memory sockets and the size of the memory sticks.
- To upgrade memory it may be necessary to remove the old memory before installing the new. Therefore the benefit is only the difference in capacity of the memory sticks.
- Latest technology high density memory sticks sell at a premium, and can exceed the cost of a system.



- The current cost of enterprise quality high performance 15K SAS disks is $1 to $4 per Gigabyte (50 times less than memory).
- Slower SATA disks (7200) are about 30 cents (167 times less than memory)
- Disks can easily be added and the old disks still used. Most systems can be expanded to 100 direct attached storage (DAS) disks.
- Further capacity can be added with a storage area networks (SAN)



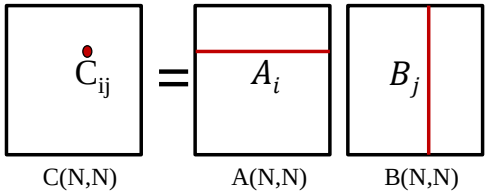
- Showing memory and disk together, it is obvious that disks provide a lower cost and more expandable storage solution.

- While memory capacity has continued to grow, disk capacity and performance has also grown.

- Most computers are not used to compute but to store and retrieve information, including the Internet. These applications have contributed significantly to the interest in improved disk technology.

- Over the last 30 years the ratio of memory price to disk price has remained about 100 to 1.

Data Reuse: The Matrix Multiply



$C_{ij} = \sum_{k=1}^{N} A_{ik} B_{kj}$

M=Memory(Bytes)
M=(8 bytes/word)(5 matrices)N^2

$N = \sqrt{M/40}$

	Real	Complex
Ops to Multiply	$2N^3$	$8N^3$
Words to Transfer	$2N^2$	$4N^2$
Reuse per Word	N	2N
Time to Multiply	$(2N^3)/C$	$(8N^3)/C$
Time to Transfer	$(16N^2)/D$	$(32N^2)/D$
Ratio	ND/8C	ND/4C

C=Compute Rate (Flops/Sec.)
D=Transfer Rate (Bytes/Sec.)

fmslib.com

- For a matrix multiply of matrices having a dimension of N, each term requires a dot product of length N between a row of the first matrix and a column of the second.
- There are N squared terms to compute in the output matrix
- Real data requires one multiply and one add for a total of 2 N cubed operations
- Complex data requires 4 multiplies and 4 adds (due to the real and imaginary parts and their cross products) for a total of 8 N cubed operations.
- While this matrix is being computed, suppose we want to transfer in 2 more blocks, say a new [A] and [B]. The number of words that must be transferred is 2N squared for real data and 4N squared for complex data.
- Therefore each word transferred is used N (or 2 N) times.
- This property is used at all levels of matrix computation, from registers to cache to memory and to disk.

- If N is 10,000 then the disk transfer rate can be 10,000 times slower than the compute rate.
- The time required to perform the multiply is the number of floating point operations divided by the computational rate.
- The time required for the transfer is the number of bytes transferred divided by the disk transfer rate.
- The ratio of compute time to transfer time determines if the process is limited by computation or disk transfers.
- Ratios greater than 1 indicate a compute bound process. The ratio gives the additional amount of performance which can be added to the system without changing memory or disks.
- Ratios less than 1 indicate an I/O bound process. The reciprocal of the ratio gives the amount the disk transfer rate must be increased or the block size must be increased.
- Note that the memory must be increased by a factor of 4 to provide an increase in block size of a factor of 2.

Benchmark Machines

Machine	CPUs	GPUs	C Flops/Sec.	D Bytes/Sec.	N Block Size	CPU to IO Ratio
Laptop	1x4 AVX	0	55 E9	88 E6	7280	2.87
Z800	1x4 SSE4	2	641 E9	425 E6	17024	2.82
Server	2x6 SSE4	8	2,386 E9	265 E6	21504	0.60

Benchmark Movies

Z800	http://www.fmslib.com/z800-100k/Performance-0001.html
Server	http://www.fmslib.com/sm06-100k/Performance-0001.html

fmslib.com

- A HP Pavilion dv7-4295 laptop having a single 2.0 GHz quad core Sandy Bridge processor, 8 GBytes of memory and a single disk was tested. For this machine the CPU to IO ratio was 2.87. The sustained performance was 55 Gflops, which demonstrated that the disk can keep up with the processor. In addition this test also demonstrated that the Sandy Bridge processors perform 8 floating point operations per clock cycle.

- The next machine was a Z800 workstation having a single 5530 quad core processor, 2 NVIDIA C2050 GPUs, 48 GBytes of memory and 4 SAS 15K disks. For this machine the CPU to IO ratio was 2.82, which indicates that the performance will not be limited by disk IO. A movie showing the solution of 100,000 complex equations and 15,000 solution vectors is located at http://www.fmslib.com/z800-100k/Performance-0001.html

- The third machine was a server having two 6-core processors, 8 NVIDIA M2050 GPUs, 48 GBytes of memory and 3 SATA 7.2K disks. For this machine the CPU to IO ratio was 0.6, indicating that time will be spent waiting for IO. This can easily be fixed by adding more memory and/or more faster disks. A movie showing the solution of 100,000 complex equations and 15,000 solution vectors is located at http://www.fmslib.com/sm06-100k/Performance-0001.html

Further Information

For further information, visit Multipath's WEB site at

http://www.fmslib.com