

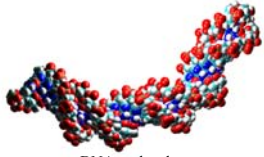


Molecular Dynamics Simulation on the Merrimac Streaming Supercomputer



Ankit Garg, Yanan Zhao (ME), Mattan Erez (CS), Abhishek Das (CS), Prof. Eric Darve (ME)

Background



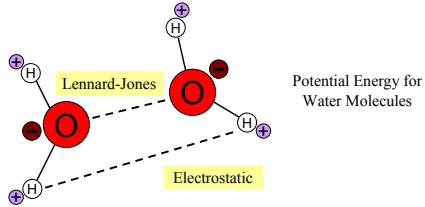
DNA molecule

Our work with molecular dynamics simulation is a preliminary step to test the performance of Merrimac on a complex scientific applications. Molecular Dynamics was chosen as it compute intensive while requiring relatively low memory bandwidth.

GROMACS and Numerical ODE in Molecular Dynamics

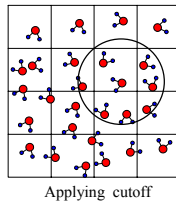
GROMACS is the most efficient MD simulation code available today. We implement the **force calculation** portion of GROMACS on a cycle-accurate Merrimac chip simulator. Our system consists of 900 water molecules.

- Newton's equations of motion $m_i \frac{d^2 \vec{r}_i}{dt^2} = -\nabla_{\vec{r}_i} U(\vec{r}_1, \dots, \vec{r}_N) = \vec{F}(\vec{r}_i)$
- Potential energy U is defined as the sum of Lennard-Jones potentials and electrostatic potentials. Those potentials are both long-range potentials.



Neighbor List Technique

- The computation of the forces $\vec{F}(\vec{r}_i)$ is easily the most expensive part of the simulation.
- To reduce its cost, all particles which are more than a **cutoff** distance apart are assumed not to interact. Particles within the cutoff radius of a central molecule form that molecule's neighbor list.



Algorithm of the Force Calculation

- Load initial input data into streams. Because of hardware constraints, all neighbor lists must be of the same length for the code to run efficiently. Consequently, molecules with extra neighbors have multiple neighbor lists. Molecules with too few neighbors are given dummy neighbors.

A. Molecule list:

0	0	1	2	2	2	899
---	---	---	---	---	---	-----	-----	-----

 Dummy molecule index

B. Neighbor list:

5	49	759	7	632	4	900	900	4	555	38	900
---	----	-----	---	-----	---	-----	-----	-----	-----	-----	---	-----	----	-----

- Compute the forces between each molecule and the molecules in its neighbor list(s). Store the forces in a new stream.

C. Forces:

F(0)	F(0)	F(1)	F(2)	F(2)	F(2)	F(899)
------	------	------	------	------	------	-----	-----	--------

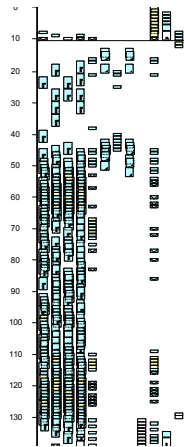
D. Scattered Forces:

F(0)	F(1)	F(2)	F(3)	...	F(899)
------	------	------	------	-----	--------

 Partial forces are combined to generate the final force stream

Optimization: The Arithmetically Intensive Kernel

The kernel performs all the arithmetic operations of the force calculation. The schedules below track the operations within one iteration of the kernel. The vertical axis marks time in number of cycles. The horizontal axis marks each functional unit in the Merrimac architecture: columns 1-4 are for addition and multiplication, column 5 is for division and square root, most of the rest of the columns are for communication. Each rectangle represents a scheduled operation.

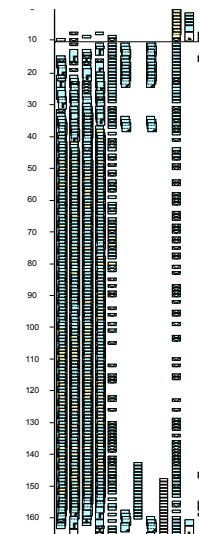


Left-hand side: No optimizations
Right-hand side: Software pipelined and loop unrolled.

- Software pipelining:** Parts of several kernel iterations that use different hardware resources and are not dependent upon each other are performed concurrently within one iteration.
- Loop unrolling:** Reduces two complete kernel iterations into one, performing most operations from both in parallel. Consequently, the schedule to the right is equivalent to two schedules from the left placed one after the other.

Result:

- The kernel is 1.7 times faster after optimization.
- 150 cycles for 18 interactions



Optimization: The Full Application

The two schedules below track the main processes in the overall application. The vertical axis marks time in number of cycles. The left column tracks chunks of the kernel currently in progress. The right column tracks I/O stream operations.

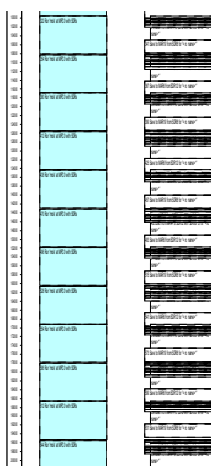


Left-hand side: automatically double-buffered.
Right-hand side: Manually stripmined, MARs increased.

- Double-buffering and Stripmining:** Our application has only one kernel, but it requires more data from input streams than can be loaded into the SRF at one time. Hence either the computer (left-hand side) or the programmer (right-hand side) splits the input streams into strips that are fed to the SRF piecemeal, causing the kernel to run piecemeal as well.
- Memory Address Registers (MARs):** channels between the memory and the SRF through which streams flow.

Result:

- no idle time between kernel calls.
- 1.35 times faster than automatically optimized code.



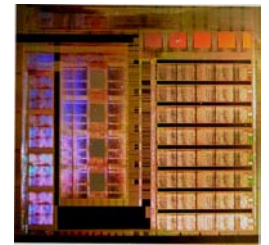
Merrimac vs. Pentium

	Run-Time (ms)	Cycles (millions)	GFLOPS
Pentium 1.7 GHz	62.0	105.4	0.563
Merrimac	1.1	1.1	21.977
Merrimac is...	56 X faster	96 X fewer	39 X more

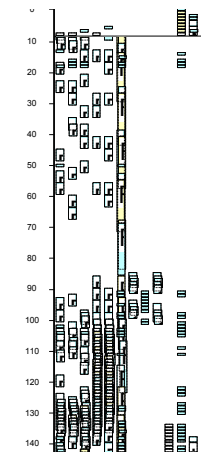
GROMACS on Hardware: Imagine



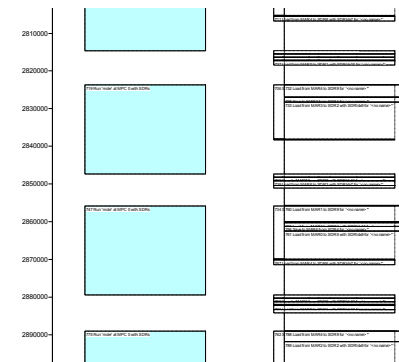
Imagine Mother Board



Imagine Processor



Kernel Schedule (software pipelined)



Application Schedule

Performance on Imagine:

- Run-time = 18 ms
- Clock speed: 200MHz
- Number of Gflops (sustained) = 1.62
- Percent of peak performance = 20.25%
- Cycles / interaction = 14 cycles
- Bandwidth from Mem/SRF/LRF = 0.8/1.264/31.92 (GB/s)