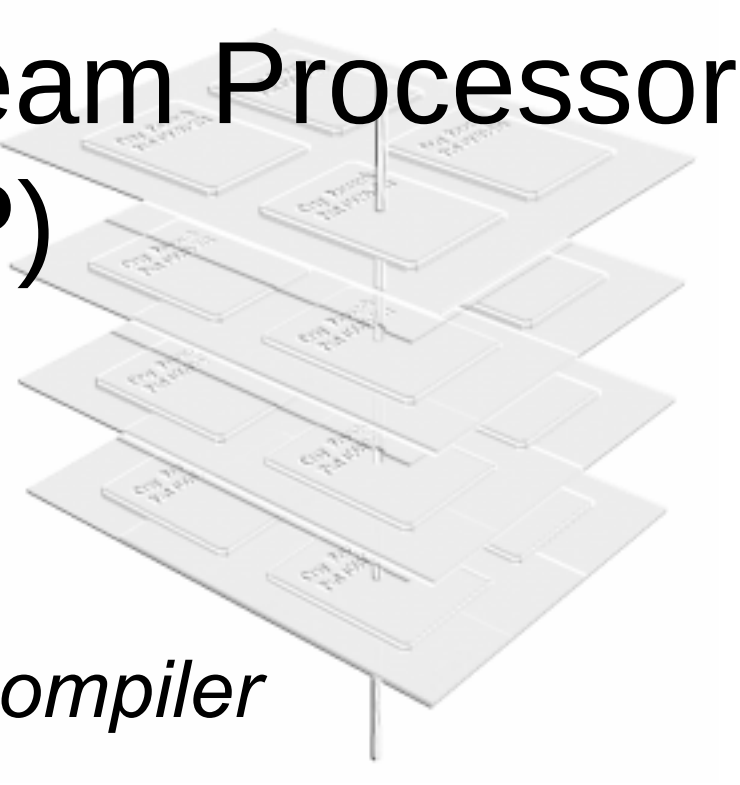
41st Cray User Group Conference
Minneapolis, Minnesota

# The SGI Multi-Stream Processor (MSP)



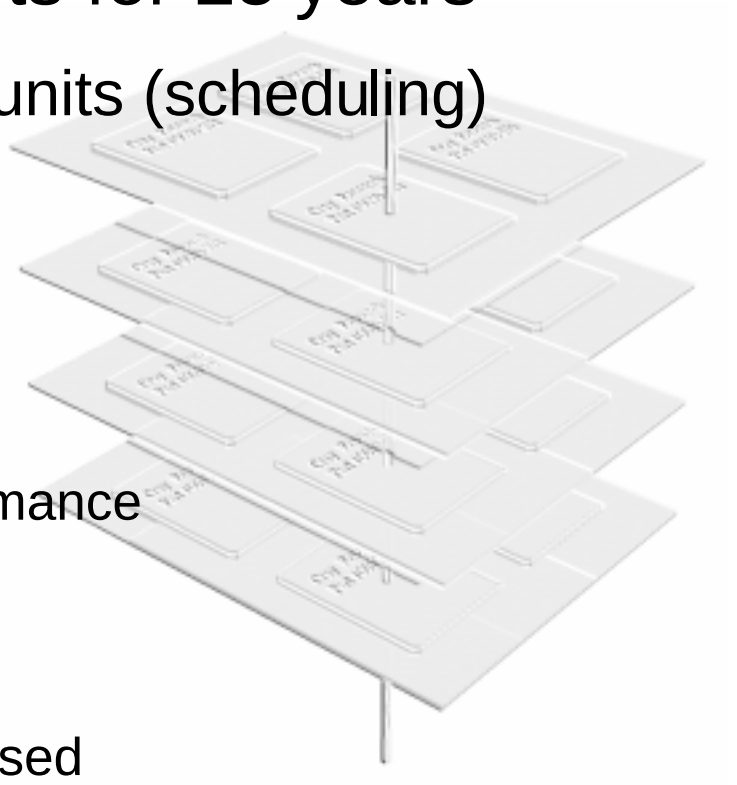
Terry Greyzck

*Technical Lead, MSP Compiler*

28-May-99

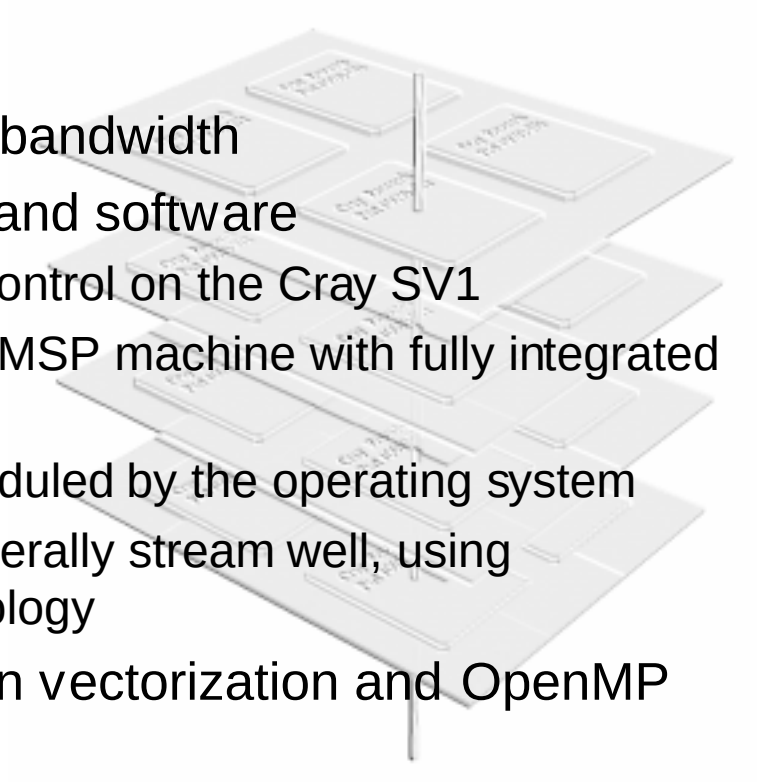
# We know parallel

- Multi-level parallelism experts for 25 years
  - Multiple, pipelined functional units (scheduling)
  - Automatic vectorization
    - Very low overhead
    - Chained, pipelined operations
    - Asynchronous loads and stores
    - High percentage of peak performance
  - Autotasking/OpenMP
    - Higher overhead
    - Both automatic and directive-based
    - Designed for maximal use of expensive processors in a time-sharing environment



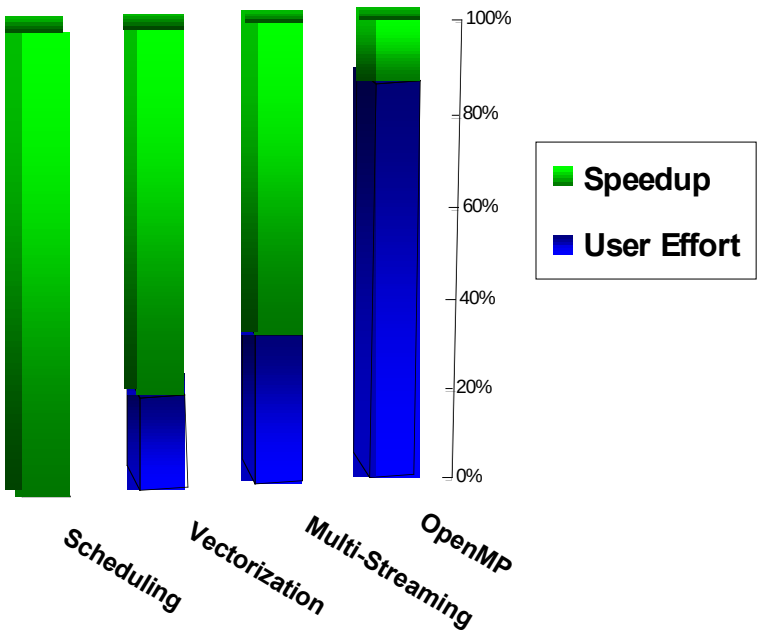
# Multi-Stream Processing

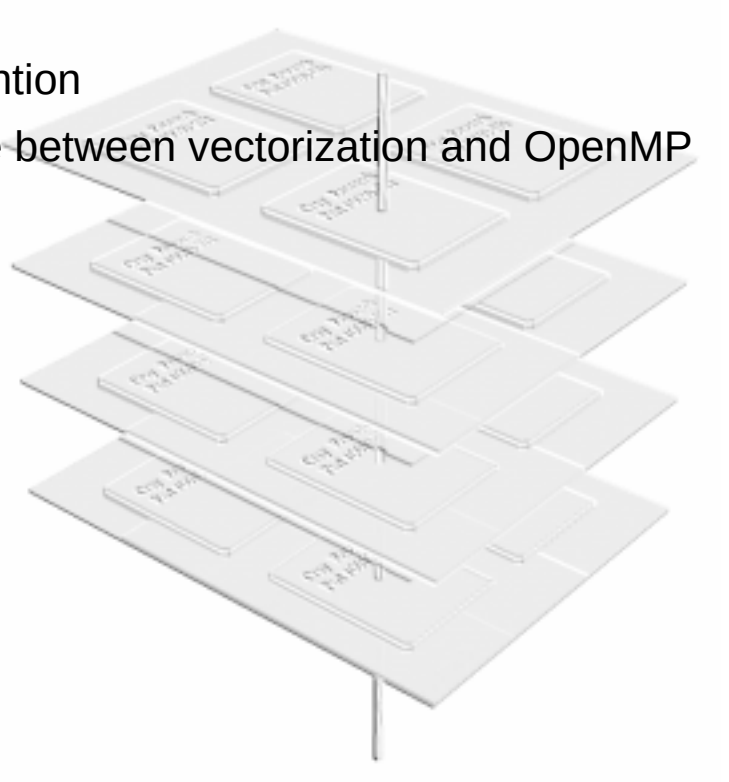
- Multi-Stream Processing
  - True parallel execution
  - Allows optimal use of memory bandwidth
  - Synergistic union of hardware and software
    - Shared registers and cache control on the Cray SV1
    - The Cray SV2 is a dedicated MSP machine with fully integrated hardware support
    - Multi-Streamed jobs are scheduled by the operating system
    - Codes that vectorize well generally stream well, using breakthrough compiler technology
  - Fills the cost/effort gap between vectorization and OpenMP



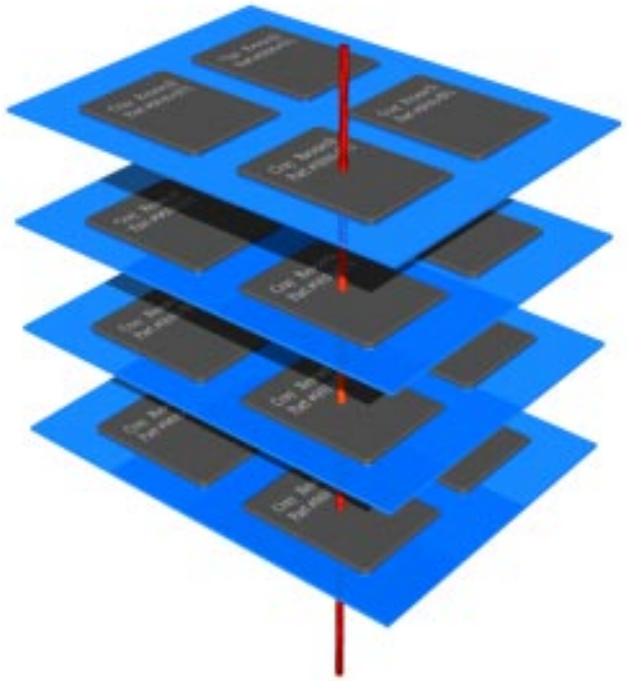
# Minimum Effort, Maximum Gain

- Manual parallelization methods require substantially more user effort than automatic vectorization or streaming
- Vectorization requires relatively little user intervention
- Multi-Streaming is designed to be a logical bridge between vectorization and OpenMP



# What's *this* thing?



It's an abstract representation of an MSP

- An SV1 board contains four SSPs
- An MSP is ideally configured by choosing one SSP from each of four boards
- The rod running through the SSPs represents a single SV1 MSP
- If you can find it on the web, it *spins*!
- The SV2 MSP configuration is very different, but I haven't created its graphic yet…
- *(yes, it will spin as well)*

# *Very* simple example…

*(Reference only)*

```
DO I = 1,1000
   A(I) = B(I) + C(I)
ENDDO
```

```
DO I = 1,250                    SSP 0
   A(I) = B(I) + C(I)
ENDDO
DO I = 251,500                  SSP 1
   A(I) = B(I) + C(I)
ENDDO
DO I = 501,750                  SSP 2
   A(I) = B(I) + C(I)
ENDDO
DO I = 751,1000                 SSP 3
   A(I) = B(I) + C(I)
ENDDO
```

- Only one version of the code exists; all SSPs execute the same loop
- Each SSP determines its own loop iteration range, based on its SSP number
- There is no ‘master’ or ‘slave’ concept; each SSP is fully independent and is capable of executing any segment of code

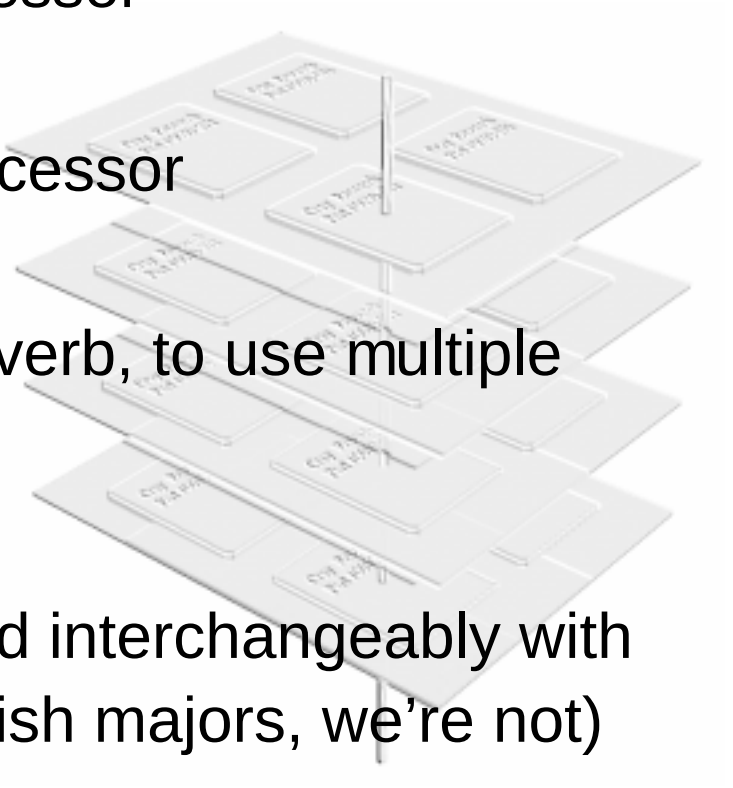
# End of the Term

MSP *(n)* Multi-Stream Processor

SSP *(n)* Single-Stream Processor

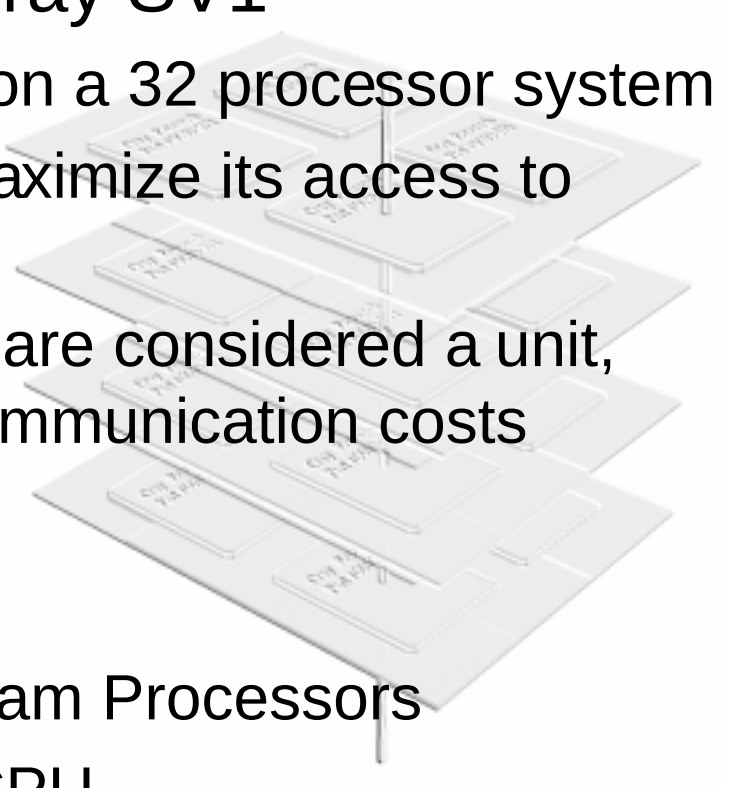
Stream *(n,v)* When used as a verb, to use multiple SSPs of an MSP

Streamed *(n,v,adj,…)* Can be used interchangeably with multi-streaming (English majors, we're not)



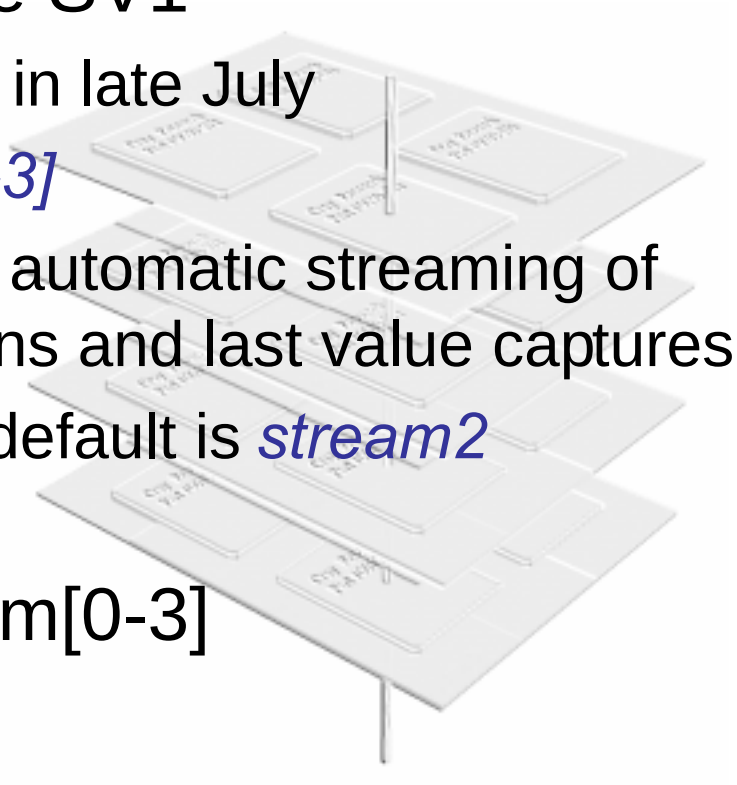
# Operating System Support

- Currently available on the Cray SV1
  - Up to six configurable MSPs on a 32 processor system
  - Each MSP is configured to maximize its access to memory bandwidth
  - All SSPs comprising an MSP are considered a unit, greatly lowering intra-MSP communication costs
- And on the Cray SV2
  - Consists entirely of Multi-Stream Processors
  - Treats an MSP like a single CPU



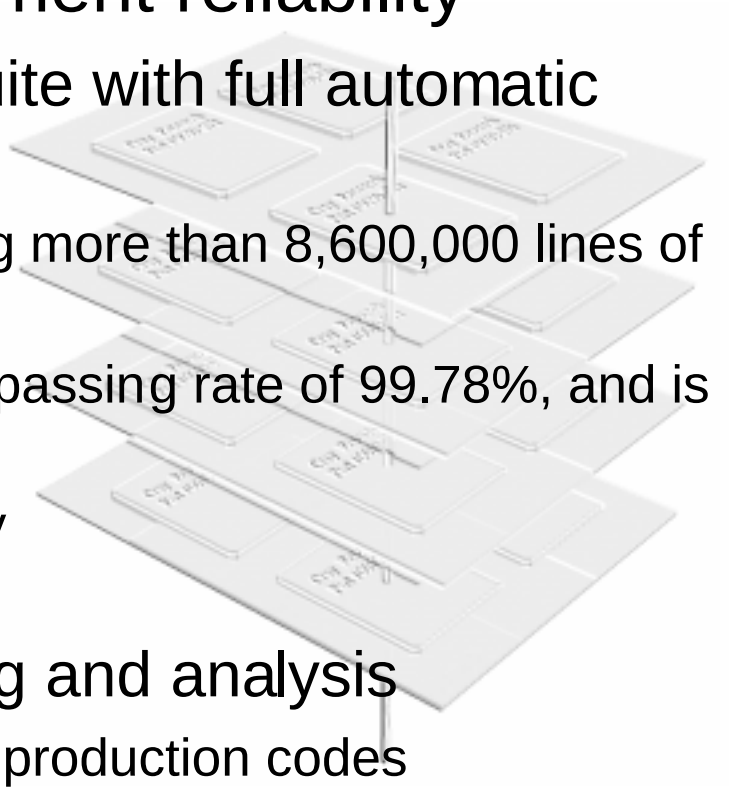
# MSP Program Support

- F90 support available on the SV1
  - Available with PE release 3.3 in late July
  - Primary option is *-Ostream[0-3]*
  - *-Ostream2* currently provides automatic streaming of loop nests, including reductions and last value captures
  - SV1 default is *stream0*, SV2 default is *stream2*
- C and C++ option is -hstream[0-3]
  - Available with PE release 3.4



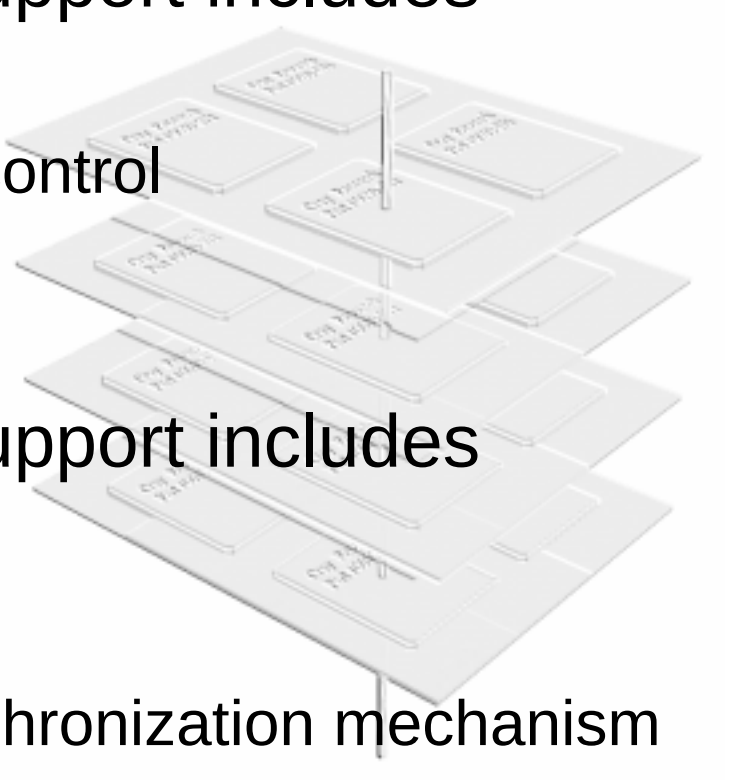
# Program Support - continued

- MSP Programming Environment reliability
  - Rigorous functional testing suite with full automatic streaming enabled
    - Over 20,000 MSP tests covering more than 8,600,000 lines of Fortran code
    - Developmental compiler has a passing rate of 99.78%, and is improving...
    - New tests being added regularly
  - Extensive performance testing and analysis
    - Of both benchmarks and actual production codes
    - Results help to channel development efforts, allowing us to focus our resources where they will do the most good



# MSP Hardware Support

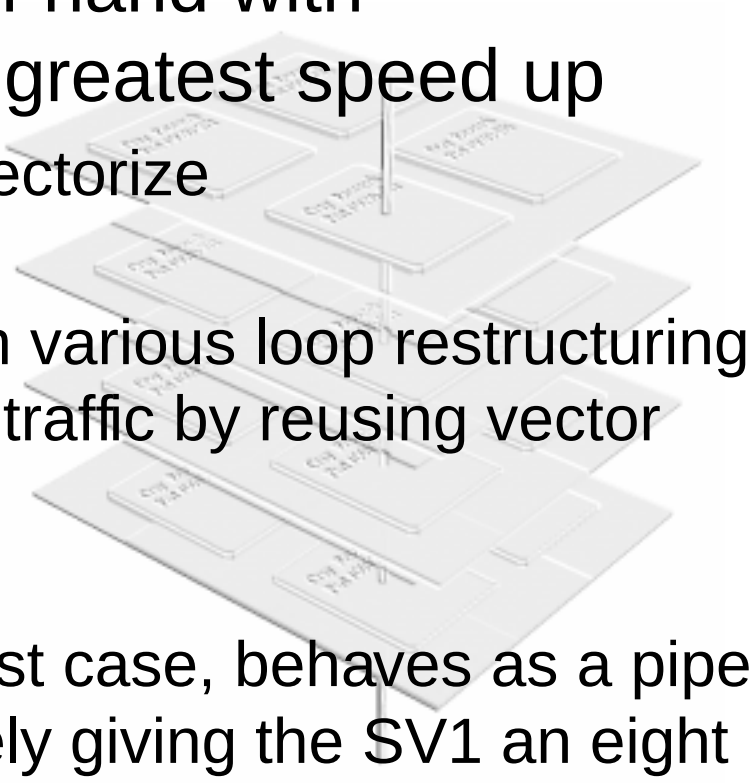
- Cray SV1 MSP hardware support includes
  - Shared primary memory
  - Intra-MSP cache coherency control
  - Shared B and T registers



- Cray SV2 MSP hardware support includes
  - Shared primary memory
  - Intra-MSP cache coherency
  - Extremely fast hardware synchronization mechanism

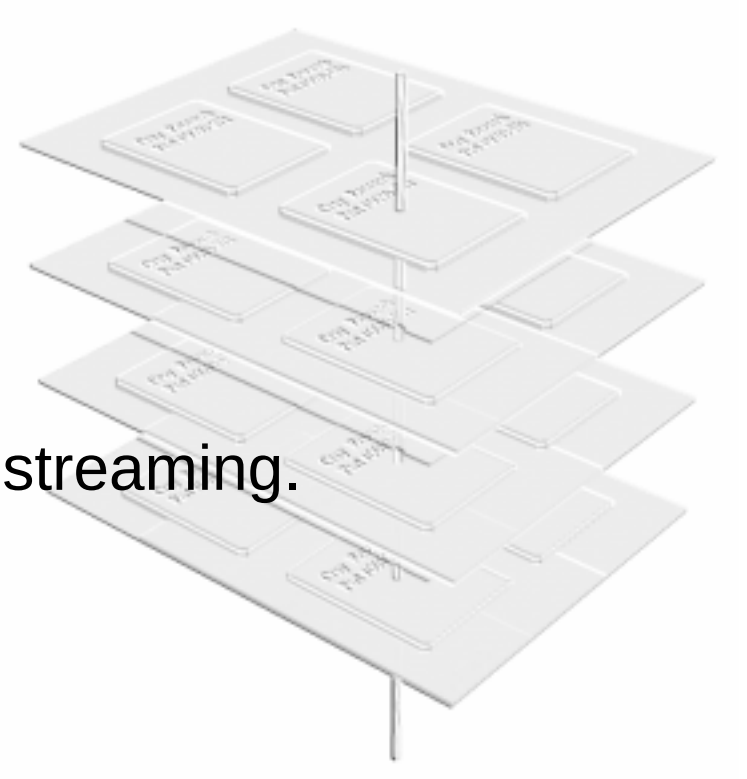
# Streaming and Vectorization

- Multi-streaming acts hand-in-hand with vectorization to achieve the greatest speed up
  - Loops can both stream *and* vectorize
  - Streaming, in conjunction with various loop restructuring techniques, reduces memory traffic by reusing vector results
  - Multi-streaming, in the simplest case, behaves as a pipe multiplier for vectors, effectively giving the SV1 an eight pipe, 256-element vector unit per MSP



# Streaming and… everything else

If it works with ***OpenMP***…

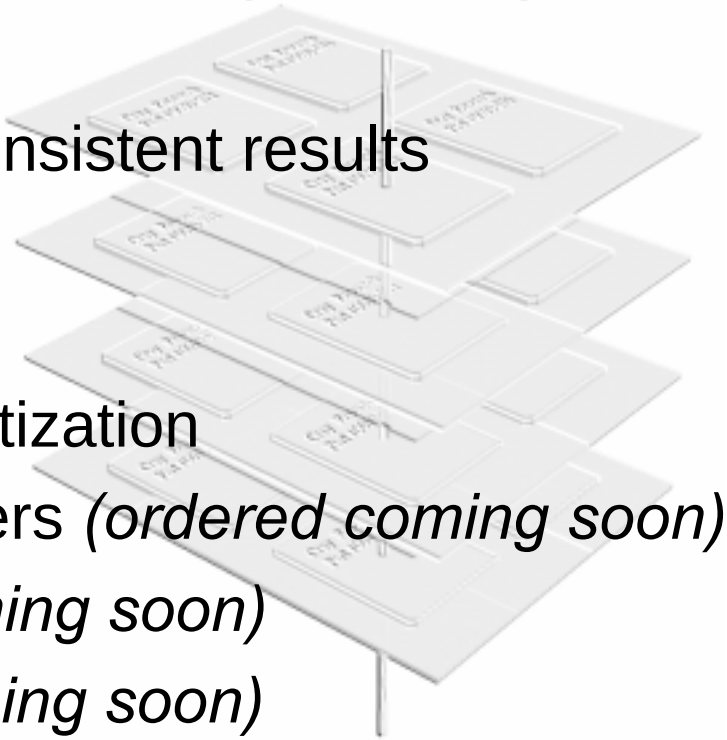


It will work with multi-streaming.

# What Multi-Streams?

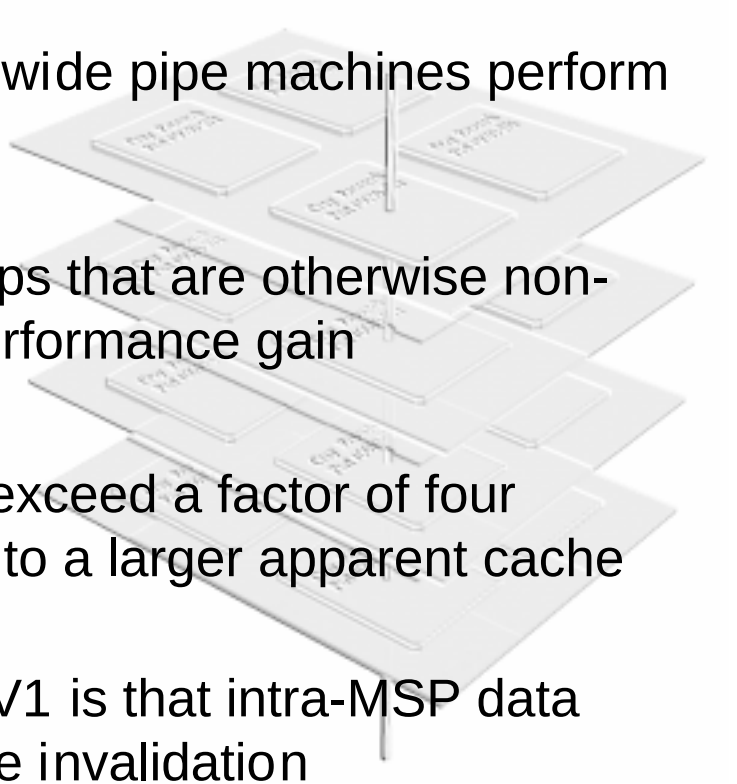
*(Reference only)*

- Inner loops, outer loops, middle loops… loops!
  - Strided loads and stores
  - Reduction operations, with consistent results
  - Conditional execution
  - Last value captures
  - Loops that require array privatization
  - Gathers and unordered scatters *(ordered coming soon)*
  - Pack/unpack operations *(coming soon)*
  - Bit matrix manipulations *(coming soon)*
  - Many other constructs available now, with more planned

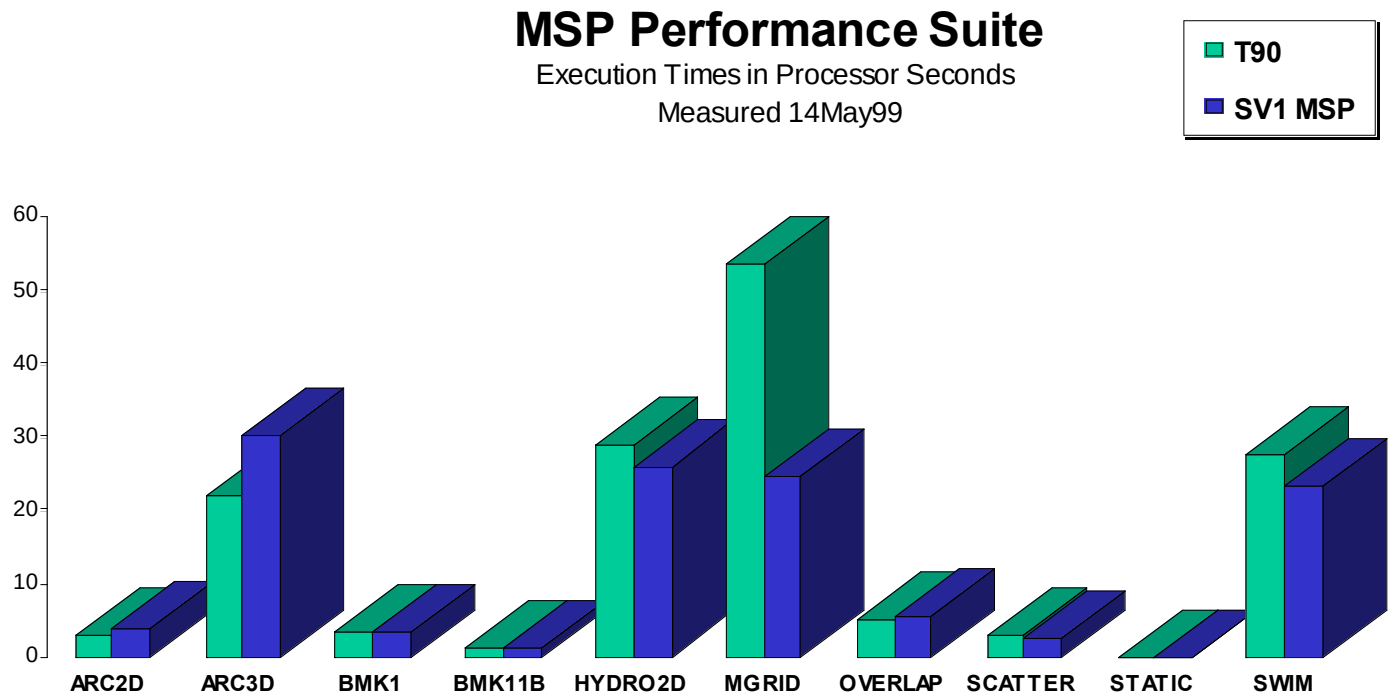


# SV1 MSP Performance

- Experiences to date
  - Codes that run well on long-vector, wide pipe machines perform very nicely using the MSP
  - The ability to multi-stream outer loops that are otherwise non-vectorizable can result in a large performance gain
  - MSP performance on the SV1 can exceed a factor of four improvement for limited cases, due to a larger apparent cache
  - The primary limiting factor on the SV1 is that intra-MSP data synchronization requires a full cache invalidation

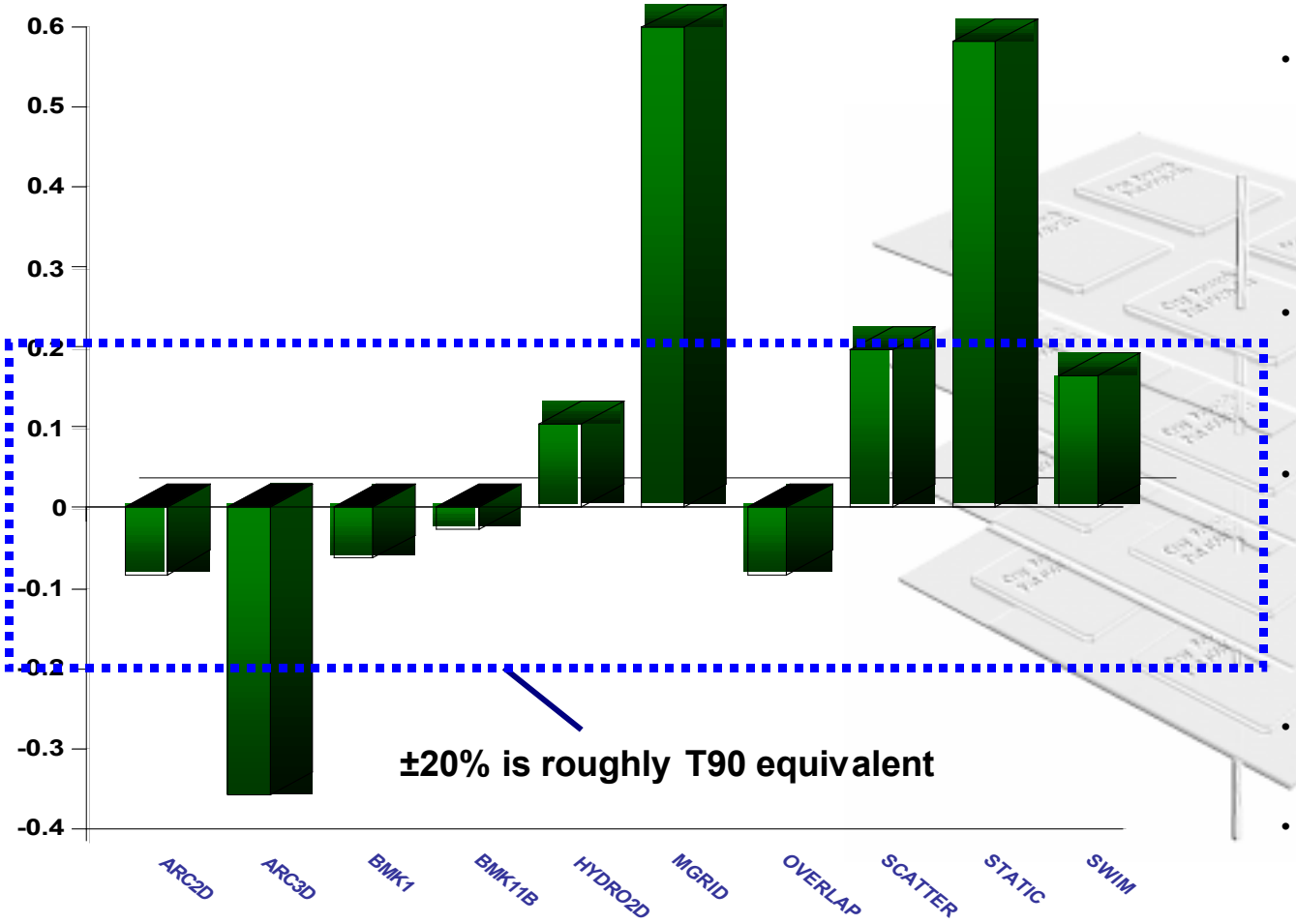


# SV1 MSP Timings




- SV1 results using a current F90 compiler with -Ostream2
- T90 results obtained using the same version of F90 with default options
- Both tests run with NCPUS = 1 (single processor execution) and no code modifications

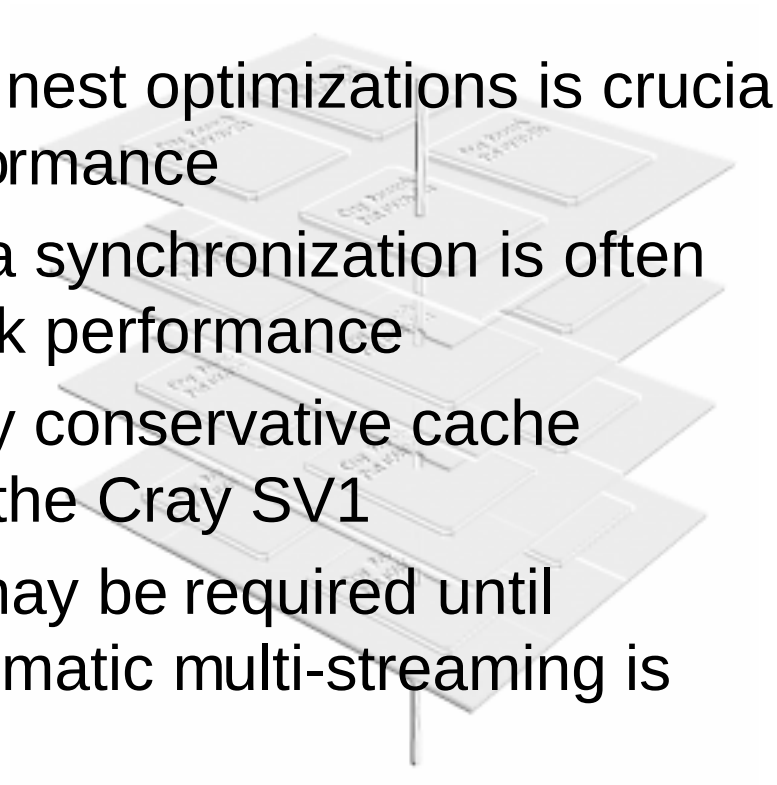
# SV1MSP / T90 Execution Speed




- Results normalilzed with respect to the T90; ratios indicate the percentage a SV1 MSP code exceeded the speed of the T90
- The range between -20% and 20% (-0.2 to 0.2) is considered to be T90 equivalent
- This comprises the MSP Performance Suite, default options at compilation except -Ostream2 for the SV1
- All tests run with NCPUS set to 1
- Y-axis is $\frac{\text{T90 - SV1MSP}}{\text{T90}}$

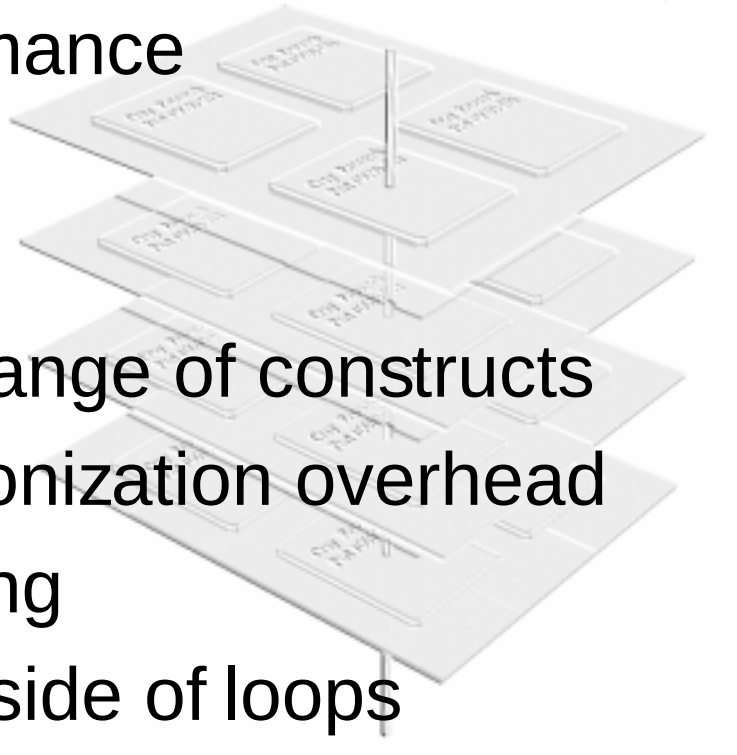
# SV1 MSP Performance Continued

- Lessons we have learned
  - The proper selection of loop nest optimizations is crucial for obtaining maximum performance
  - Reducing the amount of data synchronization is often necessary for achieving peak performance
  - Aggressive removal of overly conservative cache invalidations is essential on the Cray SV1
  - Some manual intervention may be required until compiler technology for automatic multi-streaming is fully developed



# Future Directions

- Additional directives and assertions for obtaining the highest possible performance
- Improved loop restructuring
- Improved cache utilization
- Multi-streaming of a wider range of constructs
- Further reduction of synchronization overhead
- Asymmetrical multi-streaming
- Multi-streaming of code outside of loops
- Closer interaction with inlining



# *And the winner is...*

- The multi-stream processor was designed for you, the customer
- A guiding design goal is ease of use and minimizing the necessity for hand-tuning
- At the same time, allowing low level manipulation for the seriously geeky *(that includes most of the development team, I'm afraid)*
- It's fun!  My boss told me so.
- Remember, as a company we've got 25 years of design experience behind us
- The winner is… *you*, the customer, and *us*, here at SGI.