Single-node Optimization
PGI Compiler Options

-\texttt{help} option displays command line options
  \begin{itemize}
  \item Request information about either a single option or groups of options
    \begin{verbatim}
    \% pgf90 -help=groups
    \end{verbatim}
  \item Display the available optimization switches
    \begin{verbatim}
    \% pgf90 -switch -help
    \end{verbatim}
  \end{itemize}

\begin{itemize}
  \item \begin{verbatim}
    \% pgf90 -fast -help
  \end{verbatim}
  \item Reading rcfile
    \begin{verbatim}
    /opt/pgi/6.2.2/linux8664/6.2/bin/.pgf90rc  -fast
    \end{verbatim}
  \item Common optimizations; includes \begin{verbatim}
    -O2 -Munroll=c:1
    -Mnoframe -Mlre
  \end{verbatim}
\end{itemize}
## PGI Compiler Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-fastsse</code></td>
<td><strong>==</strong> <code>-fast</code></td>
</tr>
<tr>
<td><code>-O2</code></td>
<td>This level performs all level-one local optimization as well as level two global optimization</td>
</tr>
<tr>
<td><code>-Munroll=c:1</code></td>
<td>Invoke the loop unroller. This also sets the optimization level to a minimum of <code>-O2</code>. <code>c:1</code> instructs the compiler to completely unroll loops with a constant loop count less than or equal to 1</td>
</tr>
<tr>
<td><code>-Mnoframe</code></td>
<td>Don't set up a true stack frame pointer for functions; this allows a slightly more efficient operation when a stack frame is not needed, but some options override this.</td>
</tr>
<tr>
<td><code>-Mlre</code></td>
<td>Enable loop-carried redundancy elimination</td>
</tr>
<tr>
<td><code>-Mautomainline</code></td>
<td>Inline functions with the inline attribute up to n (default 5) levels deep</td>
</tr>
<tr>
<td><code>-Mvect=sse</code></td>
<td>Pass options to the internal vectorizer. <code>sse</code>: use SSE, SSE2, 3Dnow, and prefetch instructions in loops where possible.</td>
</tr>
<tr>
<td><code>-Mscalarsse</code></td>
<td>Utilize SSE and SSE2 instructions to perform the operations coded</td>
</tr>
<tr>
<td><code>-Mcache_align</code></td>
<td>Align unconstrained data objects of size greater than or equal to 16 bytes on cache-line boundaries</td>
</tr>
<tr>
<td><code>-Mflushz</code></td>
<td>Set SSE to flush-to-zero mode</td>
</tr>
</tbody>
</table>
## PGI Compiler Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-Mnovintrap</code></td>
<td>Do not generate vector intrinsic calls</td>
</tr>
<tr>
<td><code>-M[noprefetch</code></td>
<td>[Do not] Add prefetch instructions</td>
</tr>
<tr>
<td><code>-M[nostride0</code></td>
<td>Alternate code for a loop that contains an induction variable with a possible increment of zero</td>
</tr>
<tr>
<td><code>-M[nozerotrip</code></td>
<td>[Do not] Include a zero-trip test for loops</td>
</tr>
<tr>
<td><code>-M[nodepchck</code></td>
<td>[Do not] Assume that potential data dependencies exist</td>
</tr>
<tr>
<td><code>-Mnontemporal</code></td>
<td>Enable nontemporal move prefetching</td>
</tr>
<tr>
<td><code>-M[noreentrant</code></td>
<td>[Do not] Disable optimization that may produce non reentrant code</td>
</tr>
<tr>
<td><code>-Mextract=option</code></td>
<td>Subprogram extraction phase to prepare for inlining</td>
</tr>
<tr>
<td><code>-Mprof=option</code></td>
<td>Profile options</td>
</tr>
<tr>
<td><code>-Mlist</code></td>
<td>Creates a listing file</td>
</tr>
<tr>
<td><code>-Mipa=option</code></td>
<td>Enable and specify options for InterProcedural Analysis (IPA)</td>
</tr>
<tr>
<td><code>-Minline=levels:#</code></td>
<td>Levels of inlining, levels:10 recommended for C++</td>
</tr>
<tr>
<td><code>-Msafeptr=all</code></td>
<td>C/C++ safe pointers</td>
</tr>
<tr>
<td><code>-Minfo=option</code></td>
<td>Informational messages</td>
</tr>
<tr>
<td><code>-Mneginfo=option</code></td>
<td>Why optimizations were not done</td>
</tr>
<tr>
<td><code>-mp=nonumap</code></td>
<td>OpenMP; Supported on CNL multi-core systems only</td>
</tr>
</tbody>
</table>
### PGI Fortran Compiler Flags

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-default64</code></td>
<td>Fortran driver option for <code>-i8</code> and <code>-r8</code></td>
</tr>
<tr>
<td><code>-i8, -r8</code></td>
<td>Treats INTEGER and REAL variables in Fortran as eight bytes (Caution: Use the <code>ftn -default64</code> option to link the right libraries, NOT <code>-i8</code> or <code>-r8</code>)</td>
</tr>
<tr>
<td><code>-byteswapio</code></td>
<td>Reads big-endian files in Fortran. (Cray XT systems are little endian)</td>
</tr>
<tr>
<td><code>-Mnomain</code></td>
<td>Uses the <code>ftn</code> driver to link programs with the main program (written in C or C++) and one or more subroutines (written in Fortran)</td>
</tr>
</tbody>
</table>

**Note:** The PGI Fortran `stop` statement writes a `FORTRAN STOP` message to standard out. In a parallel application, this may not scale well. To turn off this message, use the environment variable `NO_STOP_MESSAGE`.  

10/18/2010 Cray Private
## PathScale Compiler Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-Ofast</code></td>
<td>Use optimizations selected to maximize performance. Equivalent to: <code>-O3 -ipa -OPT:Ofast -fno-math-errno -ffast-math</code></td>
</tr>
<tr>
<td><code>-O3</code></td>
<td>Turns on aggressive optimization</td>
</tr>
<tr>
<td><code>-ipa</code></td>
<td>Invokes inter-procedural analysis</td>
</tr>
<tr>
<td><code>-OPT:Ofast</code></td>
<td>Use optimizations selected to maximize performance</td>
</tr>
<tr>
<td><code>-ffast-math</code></td>
<td>Improves FP speed by relaxing ANSI &amp; IEEE rules</td>
</tr>
<tr>
<td><code>-fno-math-errno</code></td>
<td>Do not set ERRNO after calling math functions that are executed with a single instruction, e.g. sqrt. A program that relies on IEEE exceptions for math error handling may want to use this flag for speed while maintaining IEEE arithmetic compatibility.</td>
</tr>
<tr>
<td><code>-OPT:alias=restrict</code></td>
<td><strong>alias</strong> specifies the pointer aliasing model to be used. <strong>restrict</strong> specifies that distinct pointers are assumed to point to distinct, non-overlapping objects</td>
</tr>
</tbody>
</table>
## GCC Compiler Options

<table>
<thead>
<tr>
<th>Option</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-ffast-math</code></td>
<td>Sets <code>-fno-math-errno</code>, <code>-ffinite-math-only</code>, <code>-funsafe-math-optimizations</code>, <code>-fno-trapping-math</code>, <code>-fno-rounding-math</code>, <code>-fno-signaling-nans</code> and <code>fcx-limited-range</code>. This option causes the preprocessor macro <code>__FAST_MATH__</code> to be defined. This option should never be turned on by any <code>-O</code> option since it can result in incorrect output for programs which depend on an exact implementation of IEEE or ISO rules/specifications for math functions.</td>
</tr>
<tr>
<td><code>-O3</code></td>
<td>Turns on all optimizations specified by <code>-O2</code> and also turns on the <code>-finline-functions</code>, <code>-funswitch-loops</code> and <code>-fgcse-after-reload</code> options.</td>
</tr>
<tr>
<td><code>-funroll-loops</code></td>
<td>Unroll loops whose number of iterations can be determined at compile time or upon entry to the loop. <code>-funroll-loops</code> implies <code>-frerun-cse-after-loop</code>. This option makes code larger, and may or may not make it run faster.</td>
</tr>
<tr>
<td><code>-fprefetch-loop-arrays</code></td>
<td>If supported by the target machine, generate instructions to prefetch memory to improve the performance of loops that access large arrays. This option may generate better or worse code; results are highly dependent on the structure of loops within the source code. Disabled at level <code>-Os</code>.</td>
</tr>
</tbody>
</table>
Other Compiler Information

- The PGI option -Mconcur, -mprof=mpi, -Mmpi, and -Mscalapack are not supported

- Fortran interfaces can be called from a C program by adding and underscore to the respective name
  - Pass arguments by reference rather than by value
  - For example to call dgetrf(

    dgetrf_ (&uplo, &m, &n, a, &lda, ipiv, work, &lwork, &info);


SSE on the AMD Opteron

- SSE means "Streaming SIMD Extensions” that enhance the x86 instruction set
- SSE2 – for single/dual-core processors
- SSE3 – for Revision E processors (dual-core)
- SSE4a – for quad-core processors
- SSE2 enhancements include:
  - 8 new 128-bit SIMD floating-point registers that are directly addressable (XMM8-15)
    - For 32-bit variables: vector instructions with vector length 4
    - For 64-bit variables: vector instructions with vector length 2
  - 50 new instructions for packed floating-point data
  - MMX instructions for multimedia, video, mpeg
## AMD Registers

<table>
<thead>
<tr>
<th>General-purpose registers (GPRs)</th>
<th>64-bit media and floating-point registers</th>
<th>128-bit media registers</th>
</tr>
</thead>
<tbody>
<tr>
<td>RAX</td>
<td>MMX0/FPR0</td>
<td>XMM0</td>
</tr>
<tr>
<td>RBX</td>
<td>MMX1/FPR1</td>
<td>XMM1</td>
</tr>
<tr>
<td>RCX</td>
<td>MMX2/FPR2</td>
<td>XMM2</td>
</tr>
<tr>
<td>RDX</td>
<td>MMX3/FPR3</td>
<td>XMM3</td>
</tr>
<tr>
<td>RBP</td>
<td>MMX4/FPR4</td>
<td>XMM4</td>
</tr>
<tr>
<td>RSI</td>
<td>MMX5/FPR5</td>
<td>XMM5</td>
</tr>
<tr>
<td>RDI</td>
<td>MMX6/FPR6</td>
<td>XMM6</td>
</tr>
<tr>
<td>RSP</td>
<td>MMX7/FPR7</td>
<td>XMM7</td>
</tr>
<tr>
<td>R8</td>
<td></td>
<td>XMM8</td>
</tr>
<tr>
<td>R9</td>
<td></td>
<td>XMM9</td>
</tr>
<tr>
<td>R10</td>
<td></td>
<td>XMM10</td>
</tr>
<tr>
<td>R11</td>
<td></td>
<td>XMM11</td>
</tr>
<tr>
<td>R12</td>
<td>0</td>
<td>XMM12</td>
</tr>
<tr>
<td>R13</td>
<td>EFLAGS</td>
<td>XMM13</td>
</tr>
<tr>
<td>R14</td>
<td>Instruction pointer</td>
<td>XMM14</td>
</tr>
<tr>
<td>R15</td>
<td>RIP</td>
<td>XMM15</td>
</tr>
</tbody>
</table>

Flag registers: EFLAGS, RFLAGS

Instruction pointer: RIP
If you want your application to perform well, vectorize the code
- Strongly recommended for single precision arithmetic

What vectorizes
- Inner stride-1 loops
  - Problems may occur with large body loops
    - `Mvect=nosizelimit`
  - Use the PGI `-Minfo` option to recognize loops that vectorize

What does not vectorize
- Loops that have `if` statements or any indirect or non-unit stride references
- Outer loops
- Use the PGI `-Mneginfo` option to recognize loops that do not vectorize
Vectorization

- The `-Mvect` option is included as part of `-fast`
- The vectorizer scans code searching for loops that are candidates for high-level transformations, such as
  - loop distribution
  - loop interchange
  - cache tiling
  - idiom recognition (replacement of a recognizable code sequence, such as a reduction loop, with optimized code sequences or function calls)
- In addition, the vectorizer produces extensive data dependence information for use by other phases of compilation and detects opportunities to use vector or packed Streaming SIMD Extensions (SSE/SSE2)
- In 32-bit mode, the vector length is 4 with SSE2
- In 64-bit mode, the vector length is 2 with SSE2
Memory Access Patterns (L1)

- The Opteron L1 data cache line is 64 bytes (512 bits)
- Latency from the L1 cache is 3 clocks (2/cycle)
  - Any stride that is a multiple of 64 causes L1 bank conflicts
- Addresses 512*64 (or 32768 apart) come back to the same row; a refill from L2 has a latency of 11 or more clocks; a useful principle is to avoid large powers of 2
Memory Access Patterns (L2)

- The Opteron L2 cache line is 64 bytes
- Latency from the L2 cache is 11 or more clocks
- Addresses that are 1024*64 (or 65536) apart come back to the same row; latency for a refill from memory is ~150cp; a useful principle is to avoid large powers of 2
Memory Access Patterns (L1 TLB)

- Opteron L1 data TLB caches
  - 8 VPNs for 2-MB pages or larger
    - 2-MB pages are the default for Catamount
    - Not available for CNL (until release CLE 2.1)
  - 32 virtual page numbers (VPNs) for 4-KB pages
    - 4-KB pages are the only option for CNL, until CLE 2.1
    - For Catamount use: yod -small_pages
- When using 4-KB pages the L1 TLB is refilled from the L2 TLB

![Diagram of TLB structures]

Address

<table>
<thead>
<tr>
<th>VPN</th>
<th>PFN</th>
<th>PFN</th>
<th>PFN</th>
<th>PFN</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

L1 Data TLB

[0]

[1]

[2]

[31]

VPN PFN

L2 TLB

Memory
Memory Access Patterns (L2 TLB)

- The Opteron L2 data TLB caches 512 VPNs in 128 rows, 4 ways: Only for 4 KB pages!
- Address bits 18-12 index to a row in which the VPN must occur
- Addresses that are 512 KB apart wrap to the same row
Single-core Opteron Caches

- **L1 TLB**
- **64 KB L1 Instruction cache**
- **64 KB L1 data cache**
- **L1 TLB**
  - 8 2-MB page table entries
  - 32 4-KB page table entries
- **L2 TLB**
  - 512 4-KB page table entries
  - Only available with small pages
  - Small pages was an option with Catamount and the default with CNL
- **1 MB L2 cache**
  - Shared instruction and data cache
  - L1 and L2 caches are mutually exclusive
  - The L2 is a victim cache
  - Exception - controller define prefetch
- **HyperTransport**
  - 64-byte blocks on even boundaries.
  - Starting on a non block boundary causes two HyperTransport references
One Core of a Dual-core Opteron

- 64 KB L1 Instruction cache
- 64 KB L1 data cache
- 1 MB L2 cache
  - Shared instruction and data cache
  - L1 and L2 caches are mutually exclusive
  - The L2 is a victim cache
- L1 TLB
  - 8 2-MB page table entries
  - 32 4-KB page table entries
- L2 TLB
  - 512 4-KB page table entries
  - Only available with small pages
  - Small pages was an option with Catamount and the default with CNL
- L2 is no longer used for controller detected prefetch
- HyperTransport: 64-byte blocks on even boundaries.
  - Starting on a non block boundary causes two HyperTransport references
- DDR2 memory
  - Memory interleaving
  - (Both banks must have matched DIMMs)
One Core of a Quad-core Opteron

- **L1 Instruction TLB**: 32 4-KB page table entries, 16 2-MB page table entries
- **L2 Instruction TLB**: 512 4-KB page table entries
- **L1 data TLB**: 48 4-KB and 2-MB page table entries
- **L2 TLB**: 512 4-KB and 128 2-MB page table entries

**512-KB L2 cache**
- Shared instruction and data cache
- L1 and L2 caches are mutually exclusive
- The L2 is a victim cache

**2-MB L3 cache**: shared instruction and data cache between the 4 cores. The L3 cache is “mostly” exclusive and is a victim cache for L2s. To optimize for multiple cores reading the same data the local cores can make copies of the L3 data

**HyperTransport**: 64-byte blocks on even boundaries. Starting on a non block boundary causes two HyperTransport references

**XT4 DDR2 memory**
- Memory interleaving
  - (Both banks must have matched DIMMs)

**XT5 DDR2 memory**
- Dual memory controllers
- Memory interleaving
  - (Both banks have matched DIMMs)
Additional Quad-core Enhancements

- Floating-point is now 128 bits wide
  - 1 cycle per vector
    - Previous processors consumed two cycles per vector
- Out-of-order loads, ordered retirement
- Can do 2 128-bit loads or 2 64-bit stores
- Other improvements
  - Improved performance of shuffle instructions
  - Improved transfers between floating-point and general purpose registers
  - Improved transfers between floating-point registers
  - Optimization of repeated move instructions
  - More efficient PUSH/POP stack operations
  - Adaptive prefetch, automatically advances the fetch ahead distance if the demand stream catches up to the prefetch stream
  - Write combining buffers improve performance
Quad-core Cautions

- The XMM registers are 128 bits wide; partial loads can cause merge dependencies
  - Initializations that zero the unused bit do not have these merge dependencies
- Arrays that are aligned 32K bytes apart can cause cache corruption, particularly if PREFETCHNTA is used
Quad-core Information

- Stride-1 memory access patterns are important
  - Stride-1 uses all of the cache line before it can be evicted
  - The compiler often performs prefetch for stride-1 loops, not as often for non-stride-1 loops
- It is advisable to align hot loops on 32-byte blocks
- PREFETCH versus PREFETCHW
  - PREFETCHW is recommended if you expect to modify the data
  - PREFETCHW sets hints of intent to write
- Use streaming instructions instead of PREFETCHW
  - If the code will overwrite one or more complete lines with new data
  - If the new data will not be used again
- Maximum of eight outstanding cache-misses
  - References to the same cache line are merged
- Align floating point data on 16-byte boundaries
Additional XT Quad-core Information

- General module additions for quad-core
  - module load xtpe-quadcore (Depricated)
  - module load xtpe-barcelona
  - module load xtpe-shanghai
  - module load xtpe-istanbul
  - module load xtpe-mc8 (Magny-cours)
  - module load xtpe-mc12 (Magny-cours)