新しく発売されたPGIコンパイラ10.0について

NVIDIA GPGPU対応
PGIアクセラレータコンパイラの紹介

February 2010

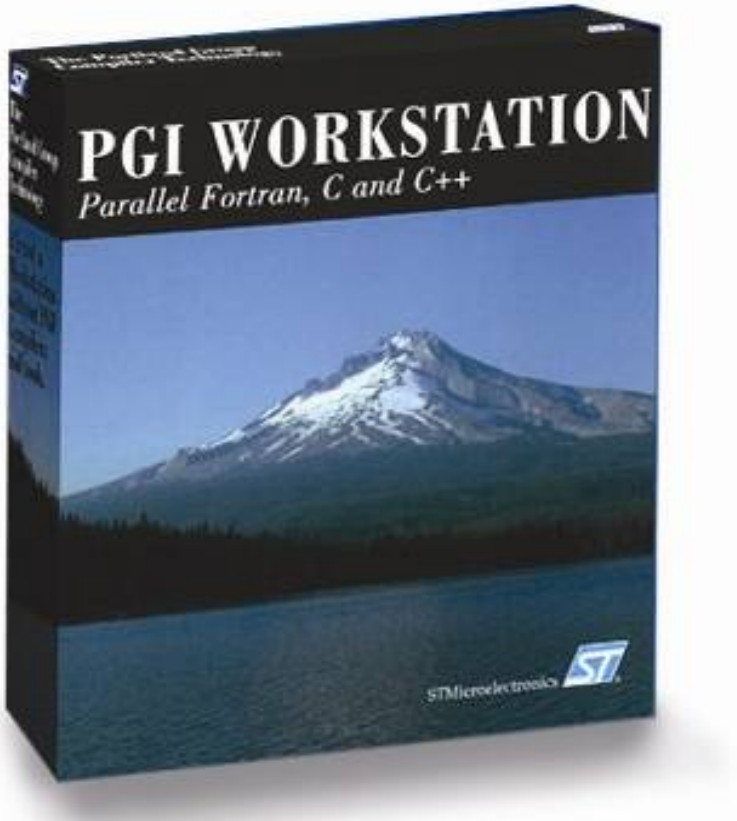
株式会社ベストシステムズ

製品紹介

PGI Workstation / Server / CDK

Linux, Windows, MacOS, 32-bit, 64-bit, AMD64, Intel 64
UNIX-heritage Command-level Compilers + Graphical Tools

Compiler	Language	Command
PGF95™	Fortran 95 w/some F2003	`pgf95`
PGCC®	ANSI C99, K&R *C* and *GNU gcc Extensions*	`pgcc`
PGC++®	ANSI/ISO *C++*	`pgCC`
PGDBG®	MPI/OpenMP debugger	`pgdbg`
PGPROF®	MPI/OpenMP profiler	`pgprof`



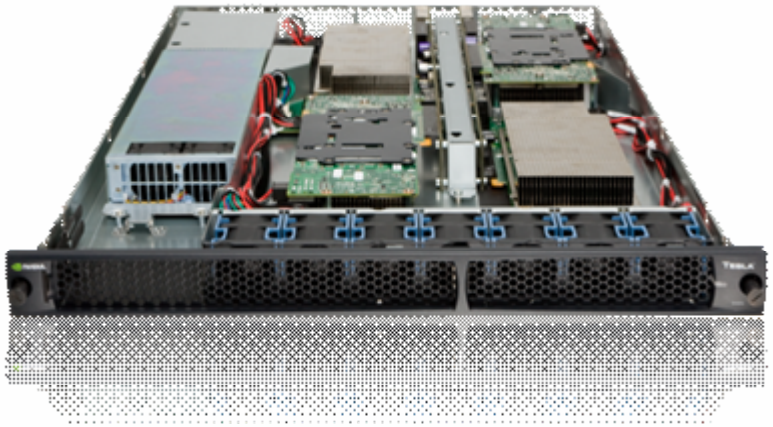
Self-contained OpenMP/MPI Development Solution

The Portland Group

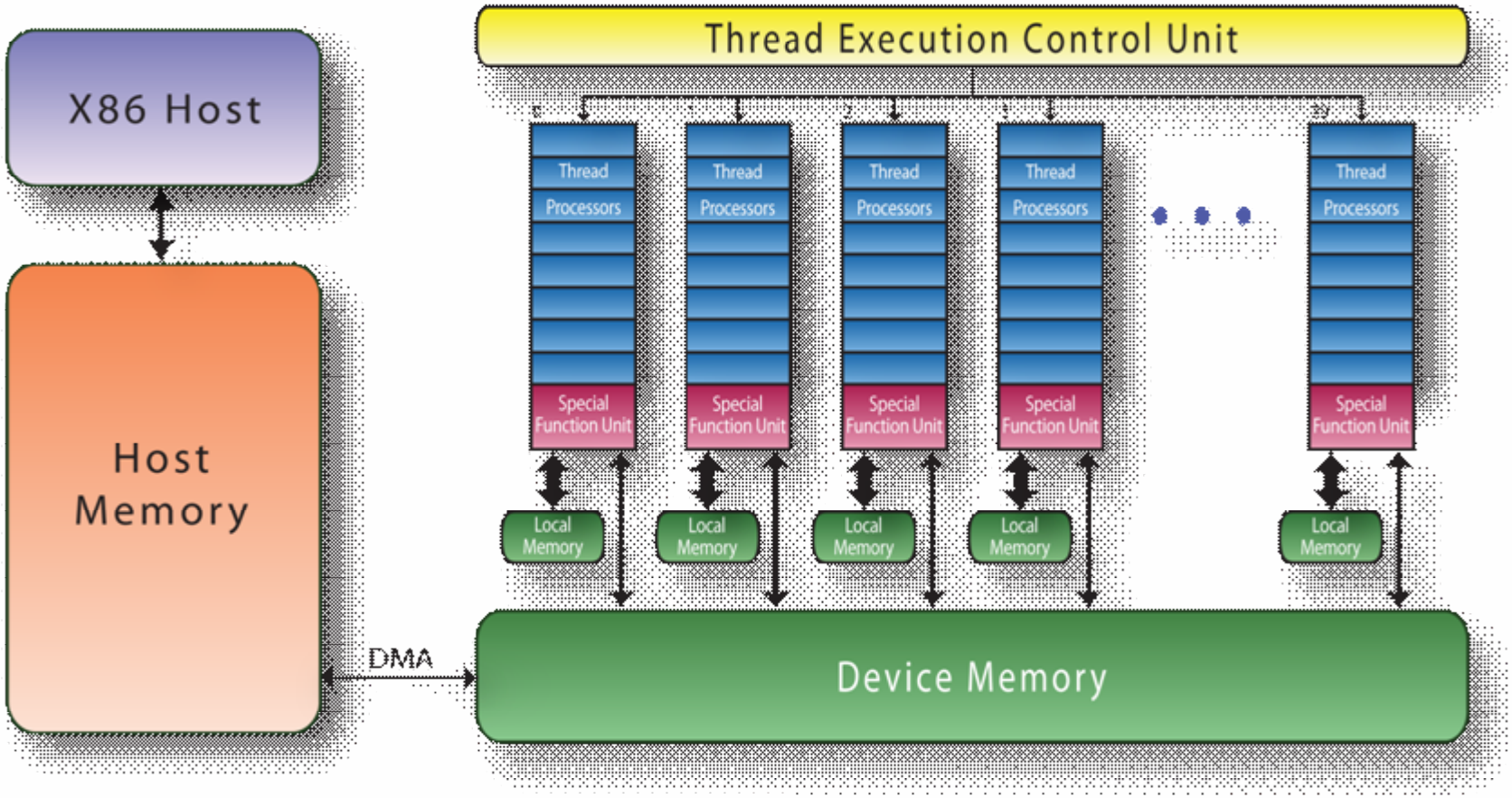
BEST SYSTEMS

GPGPU概要


NVIDIA
TESLA



Abstracted x64+Accelerator Architecture



行列積を用いた例

Simple Fortran Matrix Multiply for an x64 Host

```
do j = 1, m
  do k = 1, p
    do i = 1,n
      a(i,j) = a(i,j) +
b(i,k)*c(k,j)
    enddo
  enddo
enddo
```

Parallel Fortran Matrix Multiply for a Multi-core x64 Host

```
!$omp parallel do
  do j = 1, m
    do k = 1, p
      do i = 1,n
        a(i,j) = a(i,j) +
 b(i,k)*c(k,j)
      enddo
    enddo
  enddo
```

Basic CUDA C Matrix Multiply Kernel for an NVIDIA GPU

```
extern "C" __global__ void
mmkernel( float* a,float* b,float* c,
          int la,int lb,int lc,int n,
            int m,int p )
{
    int i = blockIdx.x*64+threadIdx.x;
    int j = blockIdx.y;

    float sum = 0.0;
    for( int k = 0; k < p; ++k )
      sum += b[i+lb*k] * c[k+lc*j];
    a[i+la*j] = sum;
}
```

```
extern "C" __global__ void
mmkernel( float* a, float* b, float* c, int la, int lb,
    int lc, int n, int m, int p ) = tx int     }
    threadIdx.x;
    int i = blockIdx.x*128 + tx;  int j = blockIdx.y*4;
    __shared__ float cb0[128], cb1[128], cb2[128],
    cb3[128];

    float sum0 = 0.0, sum1 = 0.0, sum2 = 0.0, sum3 = 0.0;
    for( int ks = 0; ks < p; ks += 128 ){
      cb0[tx] = c[ks+tx+lc*j];     cb1[tx] =
    c[ks+tx+lc*(j+1)];
      cb2[tx] = c[ks+tx+lc*(j+2)]; cb3[tx] =
    c[ks+tx+lc*(j+3)];
      __syncthreads();
      for( int k = 0; k < 128; k+=4 ){
        float rb = b[i+lb*(k+ks)];
         sum0 += rb * cb0[k];   sum1 += rb * cb1[k];
          sum2 += rb * cb2[k];   sum3 += rb * cb3[k];
        rb = b[i+lb*(k+ks+1)];
         sum0 += rb * cb0[k+1]; sum1 += rb * cb1[k+1];
          sum2 += rb * cb2[k+1]; sum3 += rb * cb3[k+1];
        rb = b[i+lb*(k+ks+2)];
         sum0 += rb * cb0[k+2]; sum1 += rb * cb1[k+2];
          sum2 += rb * cb2[k+2]; sum3 += rb * cb3[k+2];
        rb = b[i+lb*(k+ks+3)];
         sum0 += rb * cb0[k+3]; sum1 += rb * cb1[k+3];
          sum2 += rb * cb2[k+3]; sum3 += rb * cb3[k+3];
      }
      __syncthreads();
    }
    a[i+la*j] = sum0;     a[i+la*(j+1)] = sum1;
     a[i+la*(j+2)] = sum2; a[i+la*(j+3)] = sum3;
}
```

Optimized CUDA C Matrix Multiply Kernel

Host-side CUDA C Matrix Multiply GPU Control Code

```
cudaMalloc( &bp, memsize );
cudaMalloc( &ap, memsize );
cudaMalloc( &cp, memsize );

cudaMemcpy( bp, b, memsize, cudaMemcpyHostToDevice );
cudaMemcpy( cp, c, memsize, cudaMemcpyHostToDevice );
cudaMemcpy( ap, a, memsize, cudaMemcpyHostToDevice );

dim3 threads( 128 );
dim3 blocks( matsize/128, matsize/4 );
mmkernel<<<blocks,threads>>>(ap,bp,cp,nsize,nsize,
                        nsize,matsize,matsize,matsize);

cudaMemcpy( a, ap, memsize, cudaMemcpyDeviceToHost );

cudaFree( ap );
cudaFree( bp );
cudaFree( cp );
```

PGI Acceleratorの利用

PGI Directive-based Fortran Matrix Multiply for x64+GPU

```
!$acc region
      do j = 1, m
         do k = 1, p
            do i = 1,n
               a(i,j) = a(i,j) +
 b(i,k)*c(k,j)
            enddo
         enddo
      enddo
!$acc end region
```

PGI Accelerator Directives Syntax

- Accelerator region directive
  - Fortran syntax

```
!$acc region [clause [,clause]…]
```

  - C syntax

```
#pragma acc region [clause [,clause]…] new-line
{
…
}
```

- Accelerator loop mapping directive
  - Fortran syntax

```
!$acc do [clause [,clause]…]
do-loop
```

  - C syntax

```
#pragma acc for [clause [,clause]…] new-line
for-loop
```

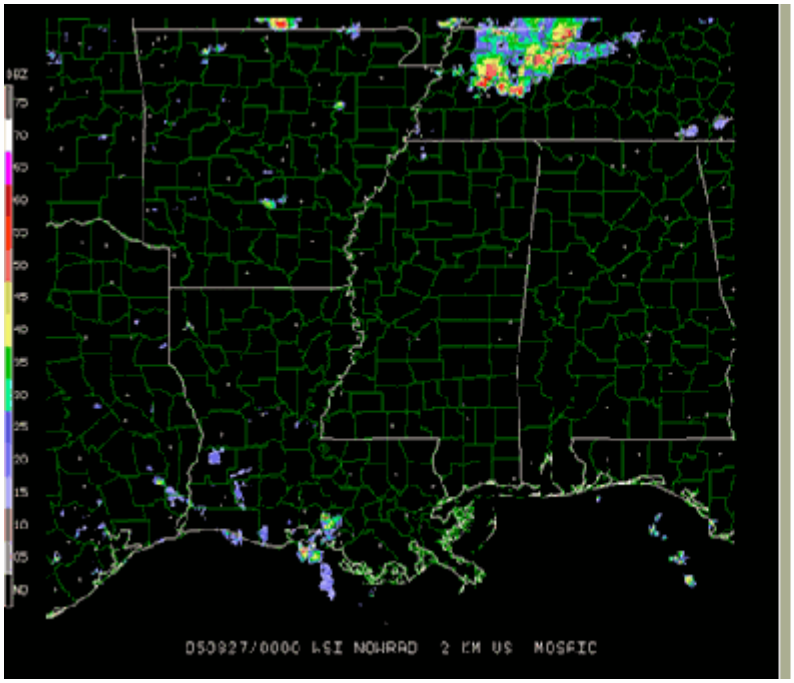
PGI Accelerator Model Advantages

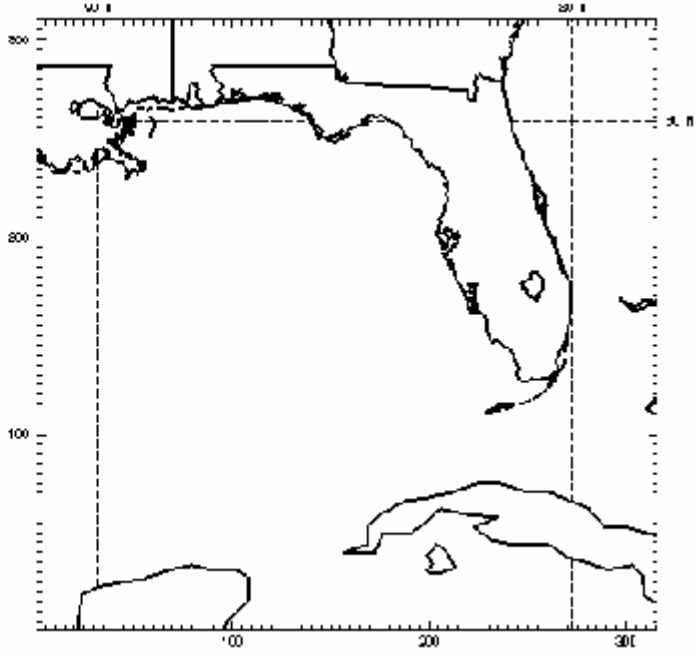
- **Minimal changes to the language** – directives/pragmas, in the same vein as vector or OpenMP parallel directives
- **Minimal library calls** – usually none
- **Standard x64 toolchain** – no changes to makefiles, linkers, build process, standard libraries, other tools
- **Not a "platform"** – binaries will execute on any compatible x64+GPU hardware system
- **Performance feedback** – learn from and leverage the success of vectorizing compilers in the 1970s and 1980s
- **Incremental program migration** – put migration decisions in the hands of developers
- **PGI Unified Binary Technology** – ensures continued portability to non GPU-enabled targets

PGI Acceleratorの実例紹介

PGI Accelerator Application Example

WRF 3.1.1 - Weather Research and Forecast Model





- Used at hundreds of sites in research and/or production
- Over 400,000 lines of Fortran & C source code
- Already MPI and OpenMP-enabled for multicore clusters
- Good candidate for a multi-core x86+NVIDIA port

```
#if ( RWORDSIZE == 4 )
#  define VREC vsrec
#  define VSQRT vssqrt
#else
#  define VREC vrec
#  define VSQRT vsqrt
#endif

!Including inline expansion statistical function
MODULE module_mp_wsm5
!
!
   REAL, PARAMETER, PRIVATE :: dtcldcr     = 120. ! maximum time step for minor loops
   REAL, PARAMETER, PRIVATE :: n0r = 8.e6         ! intercept parameter rain
   REAL, PARAMETER, PRIVATE :: avtr = 841.9       ! a constant for terminal velocity of
rain
   REAL, PARAMETER, PRIVATE :: bvtr = 0.8         ! a constant for terminal velocity of
rain
   REAL, PARAMETER, PRIVATE :: r0 = .8e-5         ! 8 microm  in contrast to 10 micro m
   REAL, PARAMETER, PRIVATE :: peaut = .55        ! collection efficiency
   REAL, PARAMETER, PRIVATE :: xncr = 3.e8        ! maritime cloud in contrast to 3.e8 in
tc80
   REAL, PARAMETER, PRIVATE :: xmyu = 1.718e-5    ! the dynamic viscosity kgm-1s-1
   REAL, PARAMETER, PRIVATE :: avts = 11.72       ! a constant for terminal velocity of
snow
   REAL, PARAMETER, PRIVATE :: bvts = .41         ! a constant for terminal velocity of
snow
   REAL, PARAMETER, PRIVATE :: n0smax =  1.e11    ! maximum n0s (t=-90C unlimited)
   REAL, PARAMETER, PRIVATE :: lamdarmax = 8.e4   ! limited maximum value for slope
parameter of rain
   REAL, PARAMETER, PRIVATE :: lamdasmax = 1.e5   ! limited maximum value for slope
parameter of snow
   REAL, PARAMETER, PRIVATE :: lamdagmax = 6.e4   ! limited maximum value for slope
parameter of graupel
   REAL, PARAMETER, PRIVATE :: dicon = 11.9       ! constant for the cloud-ice diamter
```

Porting WSM52D to NVIDIA Tesla
using PGI Accelerator Fortran vs CUDA C

- **1500 Lines of code ported**
- **3 weeks for the CUDA C Port**
- **4 days for the PGI Accelerator Fortran port**
- **5x PGI Accelerator porting efficiency advantage**
- **1.23x CUDA C performance efficiency advantage**

Using PGI Accelerator Fortran a full WRF port to NVIDIATesla GPUs is feasible in less than 1 programmer year

WRF WSM52D Microphysics Performance (Seconds)

2.67Ghz Intel Nehalem Server vs NVidia Tesla C1060

Host Cores	GPUs	Total Time	GPU Data	GPU Compute	Notes
1	**-**	236	-	-	Nehalem, 1 core
2	**-**	125	-	-	Nehalem, 2 cores
4	**-**	70	-	-	Nehalem, 4 cores
1	**1**	36.15	10.64	25.40	Initial GPU kernel
1	**1**	29.79	10.84	18.85	Tuned kernel schedule
2	**2**	16.67	6.29	10.50	Tuned kernel schedule
1	**1**	19.72	10.75	8.85	Work-in-progress kernel
2	**2**	12.00	6.87	5.29	Work-in-progress kernel
1	**1**	26.35	9.04	7.17	Hand-coded CUDA C

最後に

PGI® 2010 New Features

- PGI Accelerator™ Programming Model
  - High-level, Portable, Directive-based Fortran & C extensions (no C++, yet)
  - Supported on NVIDIA CUDA GPUs
- PGI CUDA Fortran
  - Extended PGI Fortran, co-defined by PGI and NVIDIA
  - Lower-level explicit NVIDIA CUDA GPU programming
- PVF Windows/MSMPI Cluster/Parallel Debugging
  - Debug Fortran & C MSMPI cluster applications
  - PGI Accelerator and CUDA Fortran support
- Compiler Enhancements
  - F2003 – several new language features
  - Latest EDG 4.1 C++ front-end – more g++/VC++ compatible
  - AVX code generation, code generator tuning
- PGPROF Enhancements
  - Uniform performance profiling across Linux, MacOS and Windows
  - x64+GPU performance profiling
  - Updated Graphical User Interface (GUI)

問い合わせ

- **弊社、**Bestsytems　Web**サイト**

  http://www.bestsystems.co.jp/

- **電話：03-5825-0590**
  受付時間は、平日9:00〜12:00、13:00〜17:00**となっています。**

- **電子メール：**

  sales@bestsystems.co.jp

質疑応答