スケーラブルシステムズ株式会社

OpenMPプログラミング入門
サンプルプログラムとデモ

講習の内容：Part 4

- サンプルプログラム
  - 円周率πの計算
- OmpSCRのご紹介
  - サンプルプログラム
  - OpenMPプログラムのデモ
- OpenMP最適化に関して
  - SPECOMPでの並列化と最適化の事例紹介

はじめに

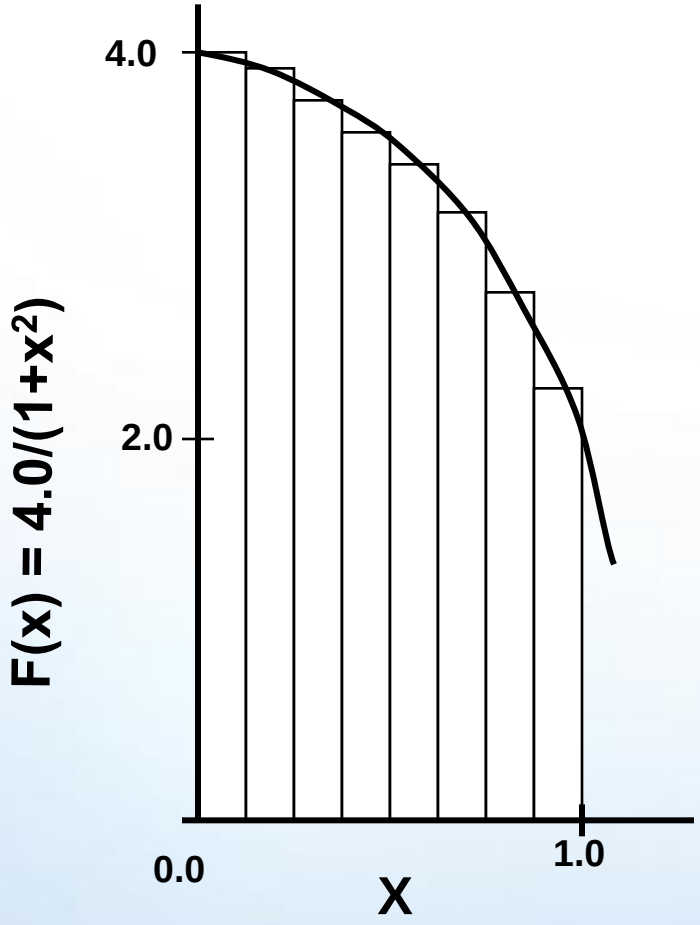
- OpenMPのサンプルプログラムは、インターネット上に数多く掲載されています。これらのサンプルプログラムを利用することで、OpenMPプログラミングについての学習は有効です。
- ただ、インターネット上のサンプルプログラムは、πの計算など比較的単純なものが多いため、より実際的なプログラミングについて、学習するには、物足りない場合もあります。
- ここでは、OpenMPのサンプルプログラムを集めた「 OmpSCR」をご紹介し、合わせてそのデモを行います。

円周率πの計算

- 単純なπの近似解を計算するプログラム
- πの計算は、並列プログラミングのサンプルでよく利用される

- πの計算のアルゴリズムには、並列処理で問題となる、ワークシェア、同期処理、リダクションなどの処理が非常に簡単なプログラムにも関わらず、含まれるため
- 計算結果の検証が非常に容易で、計算が正しく実行されたのを確認するには、良く知られた数値を確認すれば良いことも理由の一つ

円周率プログラム:数値積分



円周率πの計算式

$$\int_0^1 \frac{4.0}{(1+x^2)}\,dx = \pi$$

上記の式の積分式での近似値

$$\sum_{i=0}^{N} F(x_i)\Delta x \approx \pi$$

ここで、間隔 i の中間地点で各長方形の幅は Δx、高さは $F(x_i)$

円周率プログラム: 逐次プログラム

```
static long num_steps = 100000;
double step;
void main ()
{       int i; double x, pi, sum = 0.0;
step = 1.0/(double) num_steps;
for (i=0;i<= num_steps; i++){
     x = (i+0.5)*step;
     sum = sum + 4.0/(1.0+x*x);
}
pi = step * sum;
}
```

SPMD (Single Program Multi Data)

```
#include <omp.h>
static long num_steps = 100000; double step;
#define NUM_THREADS 2
void main ()
{         int i; double x, pi, sum[NUM_THREADS] = {0.0};
step = 1.0/(double) num_steps;
omp_set_num_threads(NUM_THREADS);
#pragma omp parallel
{      double x; int i, id;
       id = omp_get_thread_num();
       for (i=id;i< num_steps; i=i+NUM_THREADS){
            x = (i+0.5)*step;
            sum[id] += 4.0/(1.0+x*x);
       }
}
for(i=0, pi=0.0;i<NUM_THREADS;i++)pi += sum[i] * step;
}
```

円周率プログラム:ワークシェア構文(1)

```
#include <omp.h>
static long num_steps = 100000; double step;
#define NUM_THREADS 2
void main ()
{         int i; double x, pi, sum[NUM_THREADS] = {0.0};
step = 1.0/(double) num_steps;
omp_set_num_threads(NUM_THREADS);
#pragma omp parallel
{       double x; int i, id;
        id = omp_get_thraead_num();
#pragma omp for
         for (i=0;i< num_steps; i++){
                x = (i+0.5)*step;
                sum[id] += 4.0/(1.0+x*x);
         }
}
for(i=0, pi=0.0;i<NUM_THREADS;i++)pi += sum[i] * step;
}
```

円周率プログラム:ワークシェア構文(2)

```
#include <omp.h>
static long num_steps = 100000; double step;
#define NUM_THREADS 2
void main ()
{         int i; double x, sum, pi=0.0;
step = 1.0/(double) num_steps;
omp_set_num_threads(NUM_THREADS);
#pragma omp parallel private (x, sum,i)
{       id = omp_get_thread_num();
       for (i=id,sum=0.0;i< num_steps;i=i+NUM_THREADS){
             x = (i+0.5)*step;
             sum += 4.0/(1.0+x*x);
       }
#pragma omp critical
     pi += sum * step;
}
}
```

円周率プログラム:ワークシェア構文(3)

```
#include <omp.h>
static long num_steps = 100000; double step;
#define NUM_THREADS 2
void main ()
{         int i; double x, pi, sum = 0.0;
step = 1.0/(double) num_steps;
omp_set_num_threads(NUM_THREADS);
#pragma omp parallel for reduction(+:sum) private(x)
for (i=0;i<= num_steps; i++){
      x = (i+0.5)*step;
      sum = sum + 4.0/(1.0+x*x);
}
pi = step * sum;
}
```

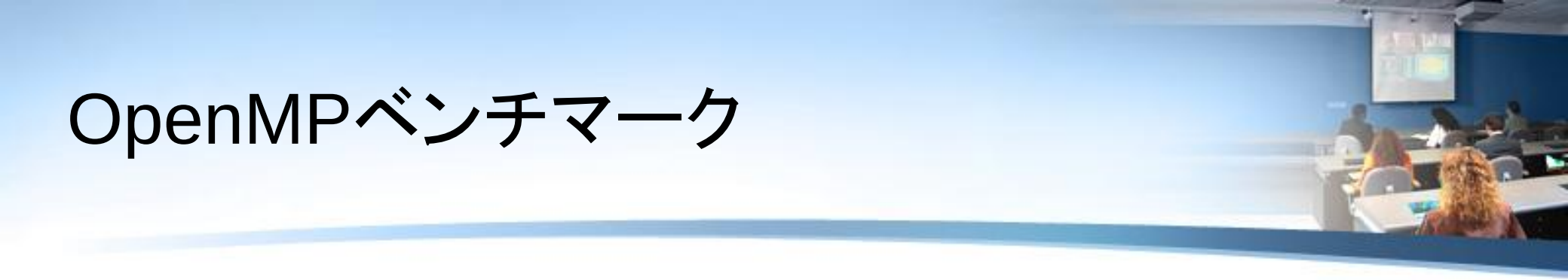
OmpSCR

OpenMP Source Code Repository

OpenMP
Source
Code
Repository

- OpenMPを利用したアプリケーションを集め、WEBで公開
- OpenMPのユーザに対して、OpenMPを利用したアプリケーションのソースを提供することで、OpenMPの利用技術の発展を図り、また、OpenMP自身の利用促進を図る
- これらのソースコードは自由にダウンロードし、利用することが可能

OpenMPベンチマーク



- SPECベンチマーク (SPEC OMP)
  - OpenMPを利用して高度に並列化されている
  - ただし、ソースコードは、非公開（有償での提供）
  - 非常に高度な並列化が適用されていて、高い並列度を示す
  - プログラミング技術や高度な並列化の習得には良いが、ソースの入手が困難
- NAS Parallel Benchmark (NPB)
  - 無償で提供されている
  - OMP/MPIなど複数の並列化APIでプログラムが提供されている
  - ソースコードが提供されているので、並列化手法などの習得には便利
  - 数値計算、特に計算流体解析（CFD)のアプリケーションとそのアプリケーションでよく使われるカーネルのベンチマークであり、数値演算に特化している
  - 高度な並列化技法が使われている

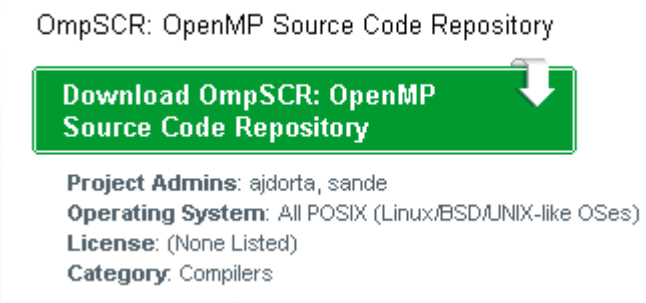
OmpSCR開発の背景

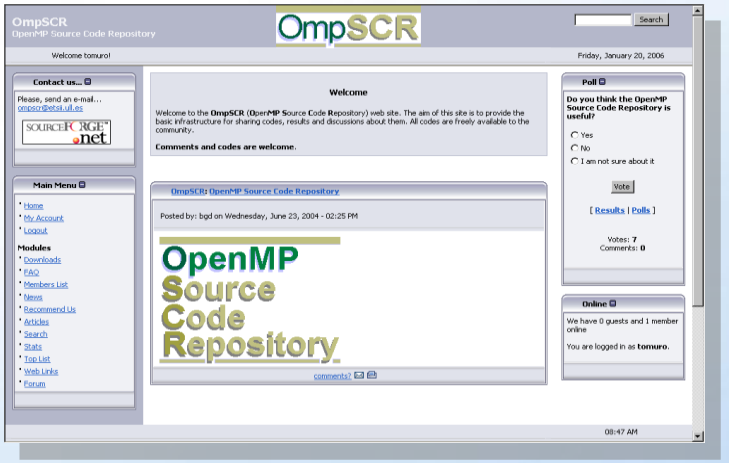
- OpenMPベンチマーク
  - SPEC OMPもNPBも有用なベンチマークプログラムではあるが、これらは、性能評価を目的として開発されている
  - OpenMPのプログラミング技術の参考にするには、よりシンプルで、また、OpenMPの動作も把握し易いアプリケーションプログラムが必要
- ベンチマーク用途以外のプログラムアーカイブ
  - ベンチマークを目的としないOpenMPアプリケーションのアーカイブの公開

ソースコードのダウンロード

- ソースコードは、soureceforge.netからダウンロード可能

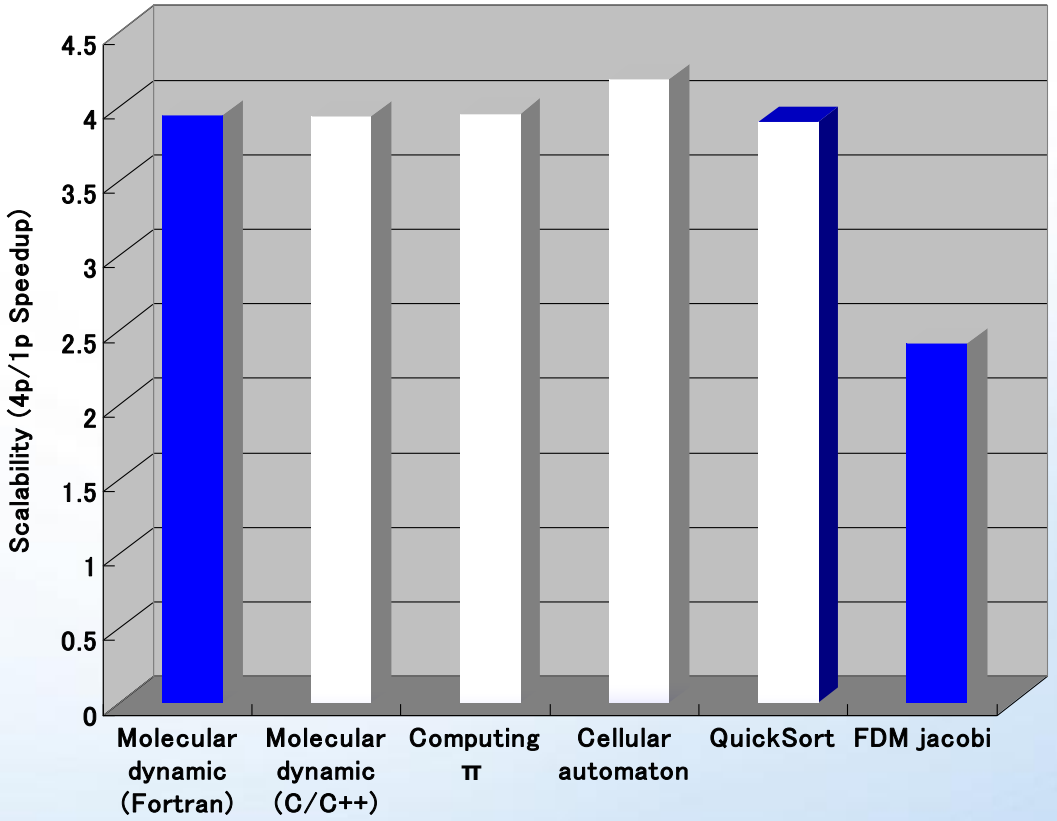
- http://sourceforge.net/projects/ompscr





OmpSCR アプリケーション

- **カーネル及びアプリケーション**
- **ANSI C/C++ 及び Fortran90/95を利用**
- **各アプリケーションの説明**
  - Computing π
  - Mandelbrot set area
  - Molecular dynamic
  - Quicksort
  - Divide and conquer fast Fourier transform
  - Bailey's "6-step" FFT
  - Loops with dependences
  - Cellular automaton
  - LU decomposition
  - Graph search
  - など



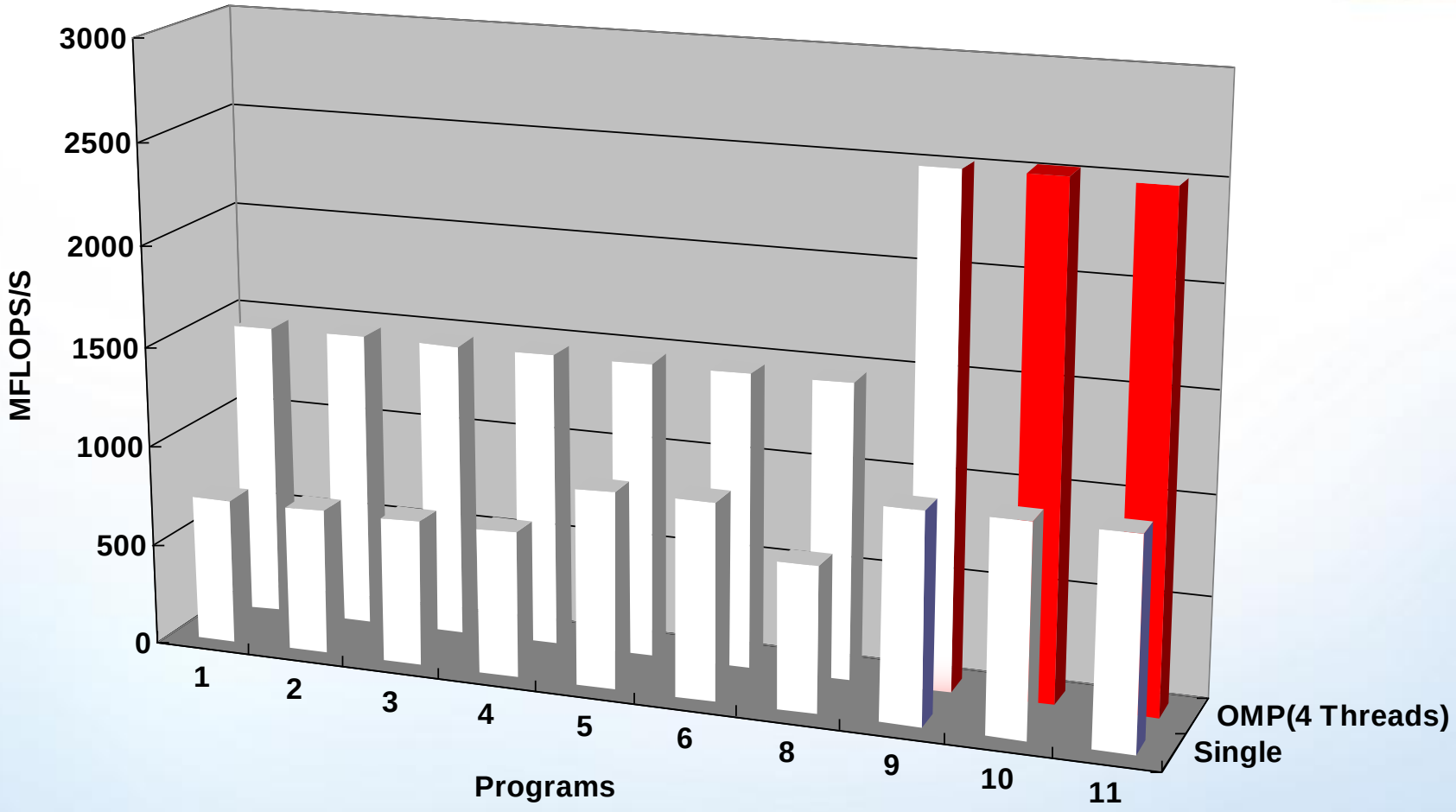
OmpSCRの利用方法

- OpenMP技術の習得での問題点
  - OpenMP APIの利用方法の資料は、WEBなどに豊富に掲載されている
  - これらのAPIの資料では、非常に簡単なループをサンプルとして示している
  - OpenMP APIの利用法のサンプルとしては、十分かもしれないが、OpenMPのプログラミング技術の習得では、さらに詳しい事例を利用できる利点は大きい
- OmpSCRの利用方法
  - シンプルなカーネルとアプリケーション

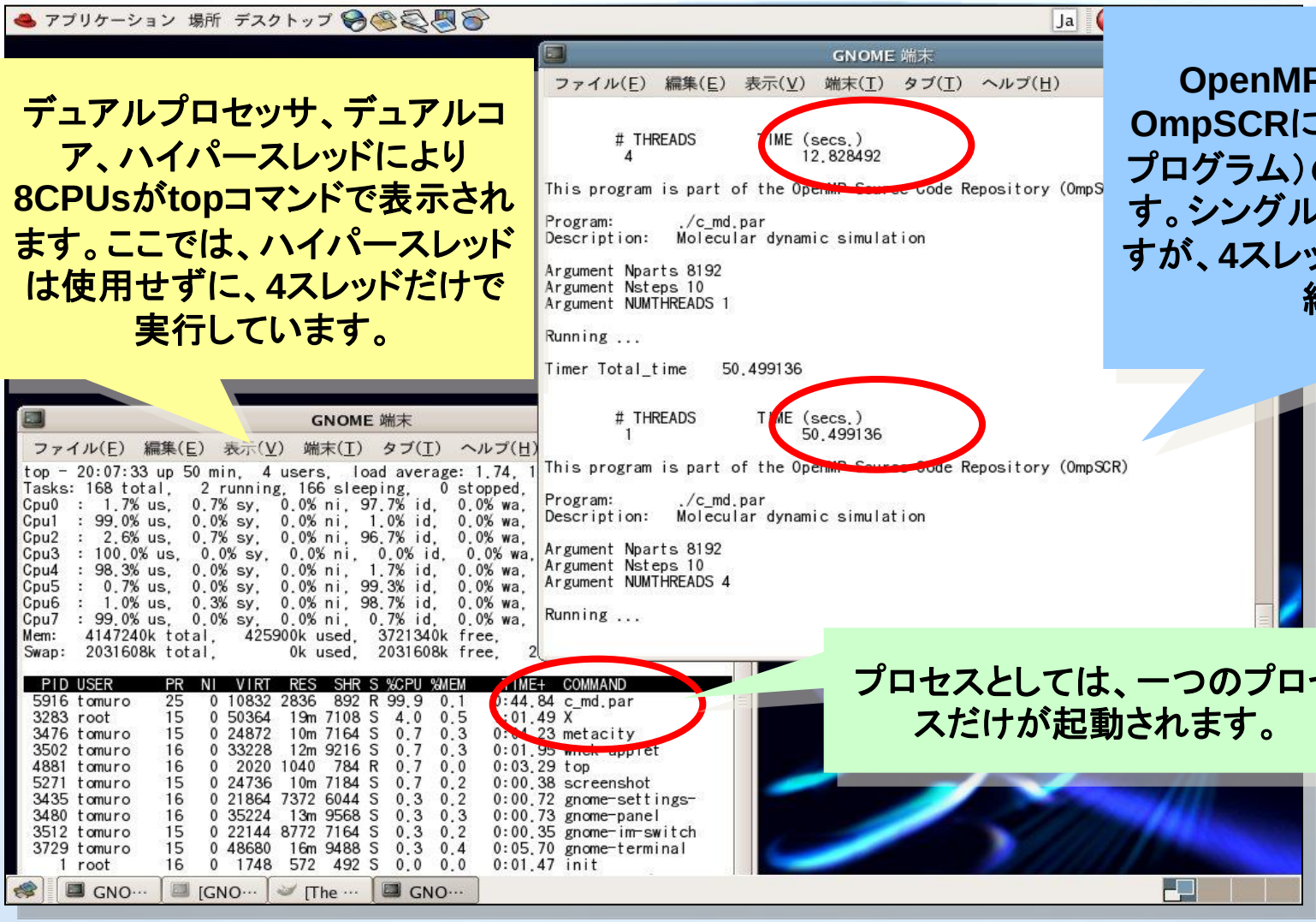
サンプル : Jacobi法によるFDM解法

f_jacobi01.f90	two parallel regions with one parallel loop each, the naive approach
f_jacobi02.f90	original OpenMP version from www.openmp.org, 2 parallel loops in one parallel region (PR)
f_jacobi03.f90	1 PR outside the iteration loop, 4 Barriers
f_jacobi04.f90	1 PR outside the iteration loop, 3 Barrier
f_jacobi05.f90	like f_jacobi04.f90, precalculation of the loop limits
f_jacobi06.f90	like f_jacobi05.f90, trying to optimize the reduction
f_jacobi08.f90	1 PR outside the iteration loop, 2-fold unrolling, software pipelining, 2 barriers per iteration
f_jacobi09.f90	1 PR outside the iteration loop, no copying, 4-fold unrolling, software pipelining, 1 barrier per iteration
f_jacobi10.f90	1 PR like V9, with errorh(0:2) to reduce code length
f_jacobi11.f90	1 PR like V9, with errorh(0:2) and uh(.,.,0:1) to reduce code length - bad performance

Jacobi法によるFDM解法



デュアルコアプロセッサ搭載サーバ



デュアルプロセッサ、デュアルコア、ハイパースレッドにより8CPUsがtopコマンドで表示されます。ここでは、ハイパースレッドは使用せずに、4スレッドだけで実行しています。

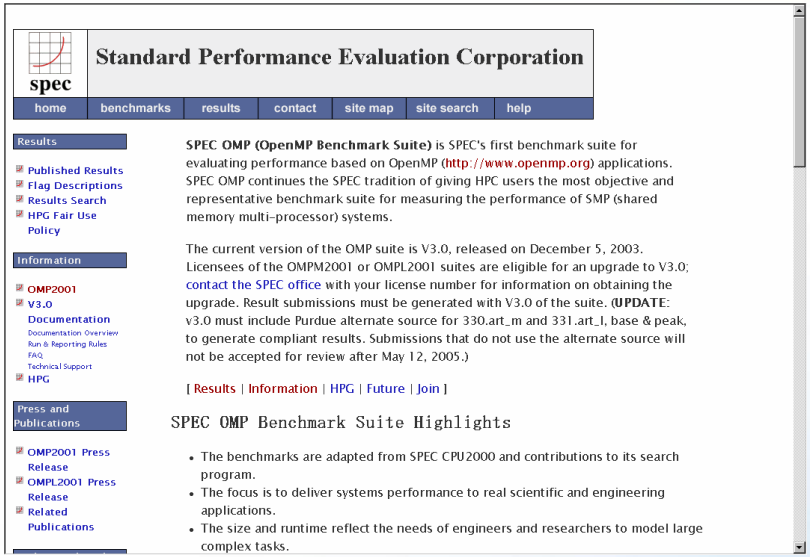
OpenMPのサンプルを集めたOmpSCRにあるMD（分子動力学-Cプログラム）のプログラムの実行例です。シングルプロセッサでは、50秒ですが、4スレッドでは、12.8秒で計算を終了出来ます

プロセスとしては、一つのプロセスだけが起動されます。

OpenMPによる並列化について
OpenMPでの最適化事例（SPEC OMP)

SPEC OMP (OpenMP Benchmark Suite) http://www.spec.org/omp/

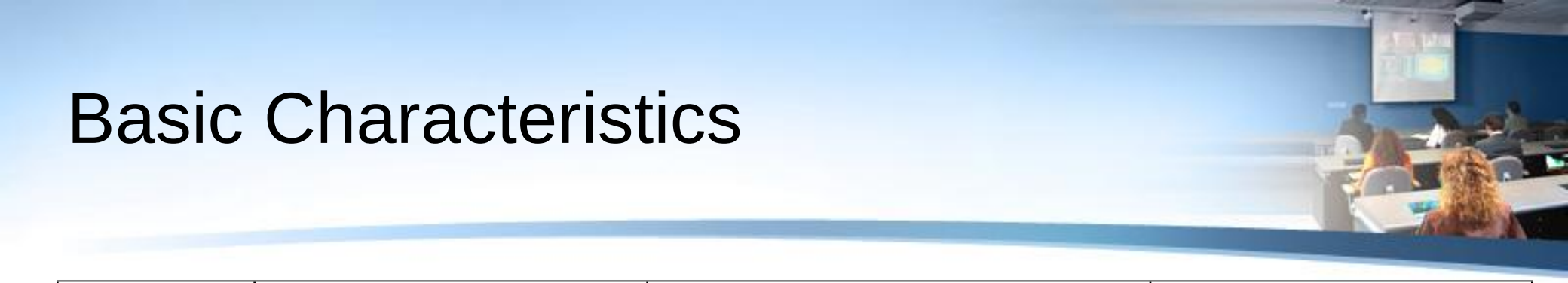
- OpenMPに関するベンチマークとしては、システムの様々なベンチマークを公開しているSPECベンチマークの一つとしてSPEC OMPがあります。SPEC　OMPでは、SPEC CPUベンチマークセットのプログラムなどを利用し、これらのベンチマークセットのプログラムをOpenMPで並列化し、その性能を評価しています。ベンチマークではMサイズとLサイズの2つの問題サイズでの性能評価が行われており、大規模SMPやNUMAシステムでは数百プロセッサまで非常に高いスケーラビリティが示されています。一つの可能性としてOpenMPのスケールアップの実証にもなっています。



SPEC OMP2001 ベンチマーク

Code	Applications	Language	lines
ammp	Chemistry/biology	C	13500
applu	Fluid dynamics/physics	Fortran	4000
apsi	Air pollution	Fortran	7500
art	Image Recognition/neural networks	C	1300
fma3d	Crash simulation	Fortran	60000
gafort	Genetic algorithm	Fortran	1500
galgel	Fluid dynamics	Fortran	15300
equake	Earthquake modeling	C	1500
mgrid	Multigrid solver	Fortran	500
swim	Shallow water modeling	Fortran	400
wupwise	Quantum chromodynamics	Fortran	2200

Basic Characteristics



Code	Parallel Coverage (%)	Total Runtime (sec)		# of parallel regions
		Single CPU	4 CPUs	
ammp	99.11	16841	5898	7
applu	99.99	11712	3677	22
apsi	99.84	8969	3311	24
art	99.82	28008	7698	3
equake	99.15	6953	2806	11
fma3d	99.45	14852	6050	92/30*
gafort	99.94	19651	7613	6
galgel	95.57	4720	3992	31/32*
mgrid	99.98	22725	8050	12
swim	99.44	12920	7613	8
wupwise	99.83	19250	5788	10

ベンチマーク：wupwise

SPEC OMPのベンチマークであるwupwiseについては、3重ループを一つのループにまとめて、並列化計算の効率化を図っています。また、このプログラムでは、大きな並列ループの中で、多くのサブルーチンが呼び出されており、大きな粒度での並列化がなされています。

```
C$OMP PARALLEL
C$OMP+        PRIVATE (AUX1, AUX2, AUX3),
C$OMP+        PRIVATE (I, IM, IP, J, JM, JP, K, KM, KP, L, LM, LP),
C$OMP+        SHARED (N1, N2, N3, N4, RESULT, U, X)
C$OMP DO
      DO 100 JKL = 0, N2 * N3 * N4 - 1
         L = MOD (JKL / (N2 * N3), N4) + 1
         LP=MOD(L,N4)+1
         K = MOD (JKL / N2, N3) + 1
         KP=MOD(K,N3)+1
         J = MOD (JKL, N2) + 1
         JP=MOD(J,N2)+1
         DO 100 I=(MOD(J+K+L,2)+1),N1,2
            IP=MOD(I,N1)+1
            CALL GAMMUL(1,0,X(1,(IP+1)/2,J,K,L),AUX1)
            CALL SU3MUL(U(1,1,1,I,J,K,L),'N',AUX1,AUX3)
            CALL GAMMUL(2,0,X(1,(I+1)/2,JP,K,L),AUX1)
            CALL SU3MUL(U(1,1,2,I,J,K,L),'N',AUX1,AUX2)
            CALL ZAXPY(12,ONE,AUX2,1,AUX3,1)
            CALL GAMMUL(3,0,X(1,(I+1)/2,J,KP,L),AUX1)
            CALL SU3MUL(U(1,1,3,I,J,K,L),'N',AUX1,AUX2)
            CALL ZAXPY(12,ONE,AUX2,1,AUX3,1)
            CALL GAMMUL(4,0,X(1,(I+1)/2,J,K,LP),AUX1)
            CALL SU3MUL(U(1,1,4,I,J,K,L),'N',AUX1,AUX2)
            CALL ZAXPY(12,ONE,AUX2,1,AUX3,1)
            CALL ZCOPY(12,AUX3,1,RESULT(1,(I+1)/2,J,K,L),1)
 100  CONTINUE
C$OMP END DO
C$OMP END PARALLEL
```

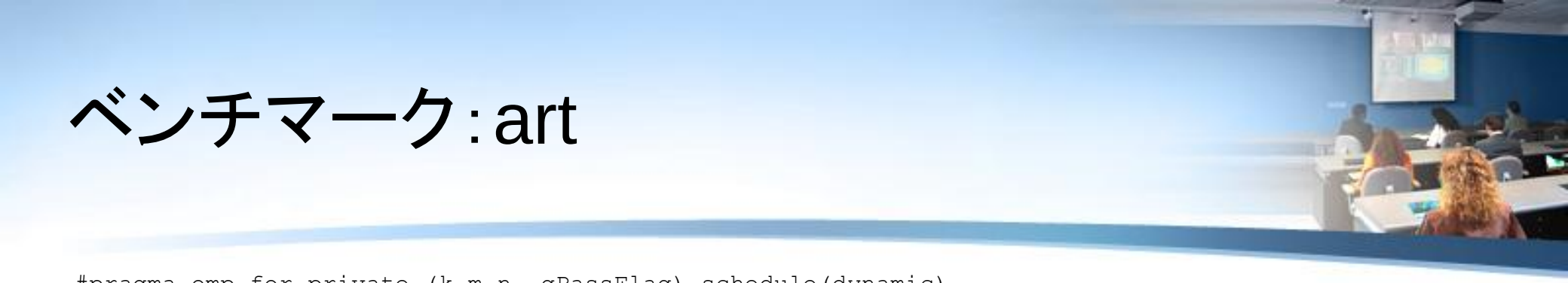
ベンチマーク：ammp

```
#pragma omp parallel for private (n27ng0, nng0, ing0, i27ng0, natoms, ii, a1, a1q, a1serial,
    inclose, ix, iy, iz, inode, nodelistt, r0, r, xt, yt, zt, xt2, yt2, zt2, xt3, yt3, zt3, xt4,
    yt4, zt4, c1, c2, c3, c4, c5, k, a1VP , a1dpx , a1dpy , a1dpz , a1px, a1py, a1pz, a1qxx ,
    a1qxy , a1qxz ,a1qyy , a1qyz , a1qzz, a1a, a1b, iii, i, a2, j, k1, k2 ,ka2, kb2, v0, v1, v2,
    v3, kk, atomwho, ia27ng0, iang0,  o ) schedule(guided)

     for( ii=0; ii<  jj; ii++)
     ...
          for( inode = 0; inode < iii; inode ++)
              if( (*nodelistt)[inode].innode > 0) {
                for(j=0; j< 27; j++)
                if( j == 27  )
     ...
                          if(  atomwho->serial > a1seri
                                for( kk=0; kk< a1->dont
                                    if( atomwho == a1->
     ...
                          for( j=1; j< (*nodelistt)[inc
     ...
                                if( atomwho->serial > a
                                    for( kk=0; kk< a1->
                                          if( atomwho =
     ...
          for (i27ng0=0 ; i27ng0<n27ng0; i27ng0++)
     ...
          for( i=0; i< nng0; i++)
     ...
              if( v3 > mxcut || inclose > NCLOSE )
     ...
```

SPEC OMPのベンチマークであるammpの計算コアの部分は、721ラインからなる大きなループです。この大きなループの並列化は可能ですが、このループ内では、if文の分岐により、ワークロードが反復計算毎に違います。そのため、静的にループを分割した場合、ワークロードの分散が不十分となり、高いスケーラビリティが得られません。このため、schedule (guided) の指定を行っています。これによって、このループのロードバランスを改善しています。

ベンチマーク：art



```
#pragma omp for private (k,m,n, gPassFlag) schedule(dynamic)
    for (ij = 0; ij < ijmx; ij++)  {
       j = ((ij/inum) * gStride) + gStartY;
       i = ((ij%inum) * gStride) +gStartX;
       k=0;
       for (m=j;m<(gLheight+j);m++)
         for (n=i;n<(gLwidth+i);n++)
           f1_layer[o][k++].I[0] = cimage[m][n];

       gPassFlag =0;
       gPassFlag = match(o,i,j, &mat_con[ij], busp);

       if (gPassFlag==1) {
          if (set_high[o][0]==TRUE) {
            highx[o][0] = i;
            highy[o][0] = j;
            set_high[o][0] = FALSE;
          }
        if (set_high[o][1]==TRUE)  {
          highx[o][1] = i;
          highy[o][1] = j;
          set_high[o][1] = FALSE;
        }
       }
    }
```

SPEC OMPのベンチマークであるartについても、schedule指示句が使われています。このコードは、画像処理プログラムであり、並列処理の効率の向上のため、2次元のforループをシングルのforループに変更しています。これによって、並列化ループの粒度を大きくしています。このコードの場合にも、計算コアの部分にif文があるため、静的なワークロードの分散は効率上問題があるため、、schedule (dynamic) の指定を行っています。これによって、このループのロードバランスを改善しています。

ベンチマーク: equake

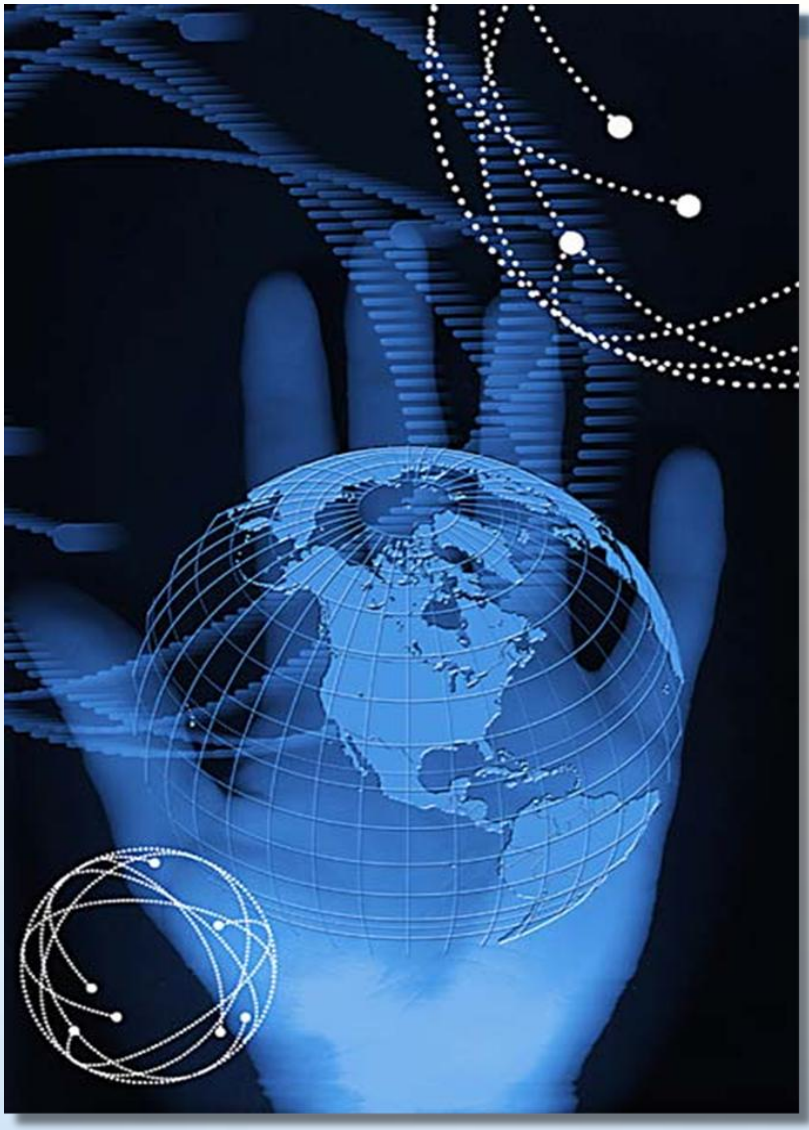
```
 /* malloc w1[numthreads][ARCHnodes][3] */
#pragma omp parallel for
  for (j = 0; j < numthreads; j++)
    for (i = 0; i < nodes; i++) { w1[j][i][0] = 0.0; ...; }

#pragma omp parallel private(my_cpu_id,exp,...)
{
  my_cpu_id = omp_get_thread_num();

#pragma omp for
  for (i = 0; i < nodes; i++)
    while (...) {
      ...
      exp = loop-local computation;
      w1[my_cpu_id][...][1] += exp;
      ...
  }
}
#pragma omp parallel for
  for (j = 0; j < numthreads; j++) {
    for (i = 0; i < nodes; i++) { w[i][0] += w1[j][i][0]; ...;}
```

各スレッド毎のローカルデータとして、配列 w1 を確保し、その配列を利用して、コアの部分の計算を行っています。このようにデータの局所化を図ることで、並列化の効率化を図ることも可能となります。

この資料について



お問い合わせ
0120-090715
携帯電話・PHSからは（有料）
03-5875-4718
9:00-18:00 (土日・祝日を除く)

WEBでのお問い合わせ
www.sstc.co.jp/contact

1/17/2010

スケーラブルシステムズ株式会社