バイオスーパーコンピューティング研究会
ウィンタースクール2014

FUJITSU
shaping tomorrow with you

コンパイラから見た将来のハードウェア

2014年1月23日
富士通株式会社
次世代テクニカルコンピューティング開発本部
言語開発統括部　第二開発部
千葉　修一

テーマ

FUJITSU

■ こんなハードウェアが欲しい

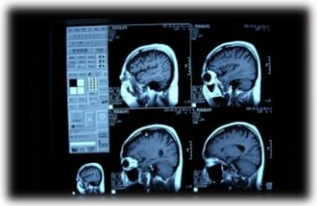
■ アプリケーションから見たハードウェアへの希望

要件の整理

- わかりやすい
- チューニングしやすい

→ 将来のハードウェア






HPCにおけるコンパイラとは

FUJITSU

■ アプリケーションとハードウェアを結びつける

「京※」、FX10などハードウェアを使いこなす

■ 最適化によりアプリケーションを高速化する

プログラムを加工し、高速なオブジェクトを生成

■ 利用者のヒントによりさらなる高速化を実現する

オプション、最適化制御行で動作が変化

* 「京」は独立行政法人理化学研究所の登録商標です。
* スーパーコンピュータ「京」は、独立行政法人理化学研究所と富士通の共同開発です。

コンパイラによる最適化とは

FUJITSU

■ ハードウェアの特性に応じた加工を行う

- ➢ 並列性を高める

  自動並列化、SIMD化、ソフトウェア・パイプライニングなど

- ➢ 命令数を削減する

  共通式の削除、ループ内不変式の移動など

- ➢ 高速な命令、および命令列に置き換える

  逆数近似命令など

asisコード → 高速なコード

分かりやすさ

FUJITSU

■ 測定プログラム

要求B/Fを12から0.5まで変化させ実行効率を比較

```
! B/F=12
real*8 a(M), b(M), c, x
do j=1, N ! parallel
   do i=1, M ! simd
      a(i) = b(i)*x+c
   enddo
enddo
```

```
! B/F=6
real*8 a(M), b(M), c, X
do j=1, N ! parallel
   do i=1, M ! simd
      a(i) = (b(i)*x+c)*b(i)+c
   enddo
enddo
```

…

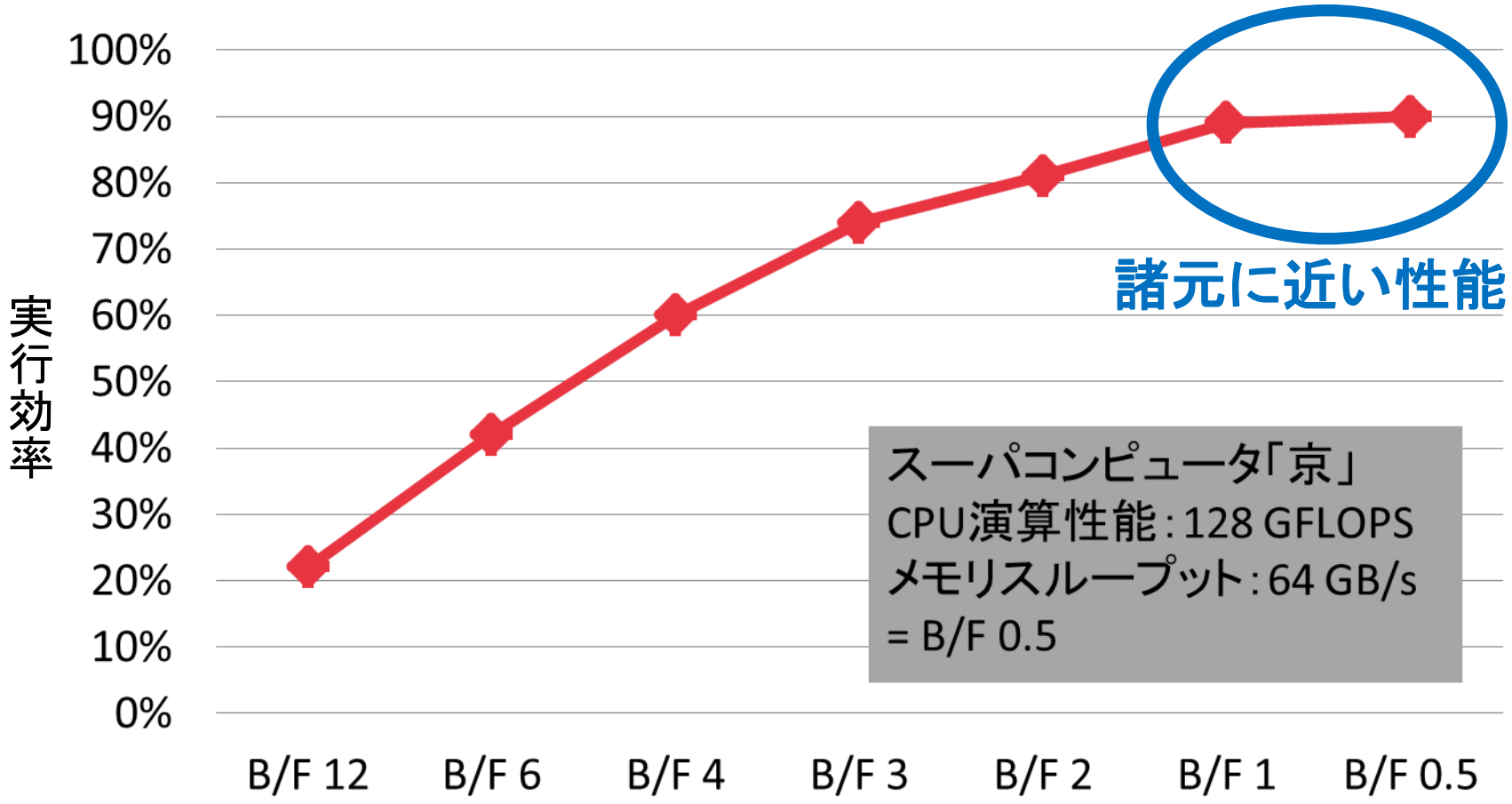
■ オプション

-Kfast,parallel

分かりやすさ

FUJITSU

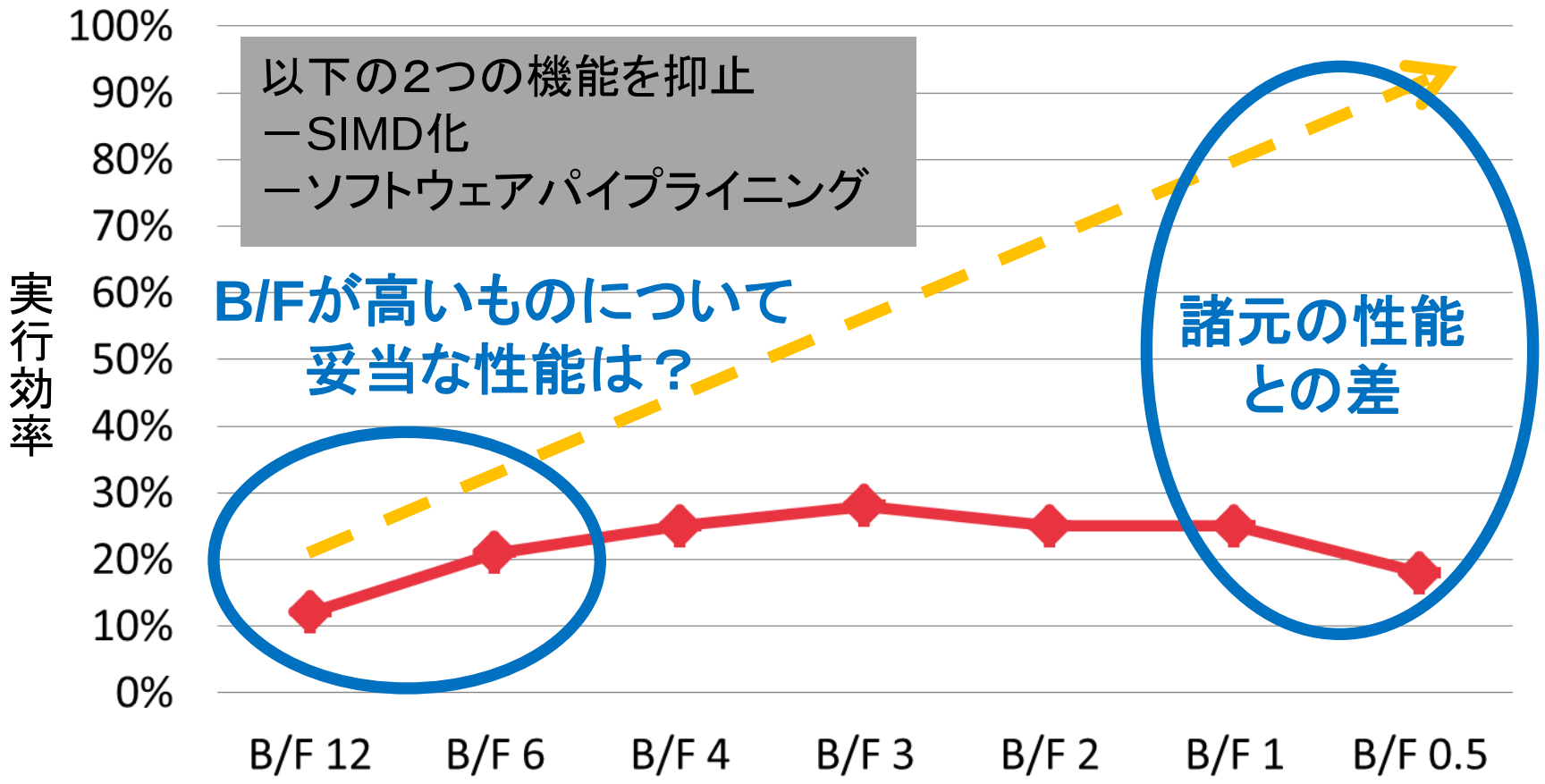
■ ハードウェアの特性が把握しやすい



分かりやすさ

FUJITSU

■ ハードウェアを使い切っていないと. . .



分かりやすさ

FUJITSU

■ ポイント

- ハードウェアを使い切っていないと、分かりやすいデータにはならない
- コードの特性によっても変化する
- 実行性能の目標はどこに設定すればいいのか？
- 課題はハードウェア？コンパイラ？

使いやすさ

FUJITSU

■ 測定プログラム

３種類の姫野ベンチマークを用意して実行性能を比較

- 配列アクセスを利用したコード
- ポインタアクセスを利用したコード
- テンプレートを利用したコード

■ オプション

-Kfast

使いやすさ

FUJITSU

■ 配列アクセスを利用したコード

```
static float  p[MIMAX][MJMAX][MKMAX];
static float  a[4][MIMAX][MJMAX][MKMAX],
              b[3][MIMAX][MJMAX][MKMAX],
              c[3][MIMAX][MJMAX][MKMAX];
static float  bnd[MIMAX][MJMAX][MKMAX];
static float  wrk1[MIMAX][MJMAX][MKMAX],
              wrk2[MIMAX][MJMAX][MKMAX];

jacobi( ... ) {
  ...
      for (...) {
        s0 = a[0][i][j][k] * p[i+1][j  ][k  ]
           + a[1][i][j][k] * p[i  ][j+1][k  ]
```

使いやすさ

FUJITSU

■ ポインタアクセスを利用したコード

```
struct mat {
  float* addr;
  int nums;
  ...
};
#define VA(mat,n,r,c,d) mat->addr[ ¥
 (n) * mat->rows * mat->cols * mat->deps + ¥
 (r) * mat->cols * mat->deps + ¥
 (c) * mat->deps + ¥
 (d) ]
jacobi( ... ) {
 ...
      for (...) {
        s0= VA(a,0,i,j,k)*VA(p,0,i+1,j,   k)
          + VA(a,1,i,j,k)*VA(p,0,i,   j+1,k)
```

使いやすさ

FUJITSU

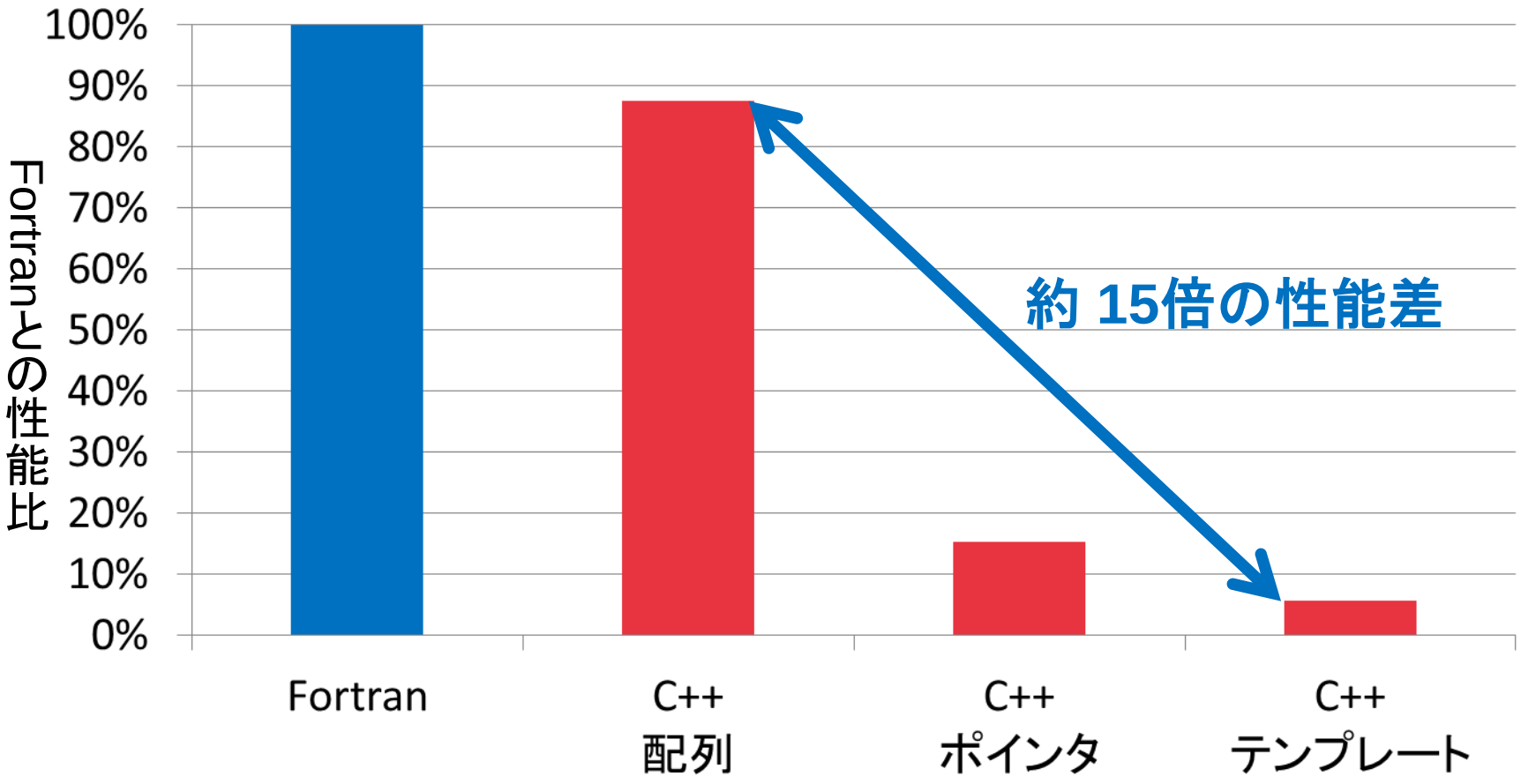
■ テンプレートを利用したコード

```
newMat(Matrix*& Mat, int nums,int rows, int cols, int deps)
{
  Mat = new (std::nothrow) boost::multi_array<float,4>(
    boost::extents[nums][rows][cols][deps] );
  return (Mat != NULL) ? 1:0;
}
main(){
  newMat(a,4, imax, jmax, kmax);
  ...
  gosa= jacobi(nn,*a,*b,*c,*p,*bnd,*wrk1,*wrk2);
}
jacobi( ... ) {
  ...
      for (...) {
        s0= a[0][i][j][k]*p[0][i+1][j][  k]
```

使いやすさ

FUJITSU

■コーディング技法による変化



使いやすさ

FUJITSU

■ 測定プログラム

最内ループの回転数を変化させ最適化の動きを比較

```
! B/F=1
real*8 a(M), b(M), c, x
do j=1, N ! parallel
   do i=1, M ! simd
      a(i) = ((((((((((b(i)*x+c)*b(i)+c)*b(i)+c)*b(i)+c)*b(i)+c)
           *b(i)+c)*b(i)+c)*b(i)+c)*b(i)+c)*b(i)+c)*b(i)+c
   enddo
enddo
```

■ オプション

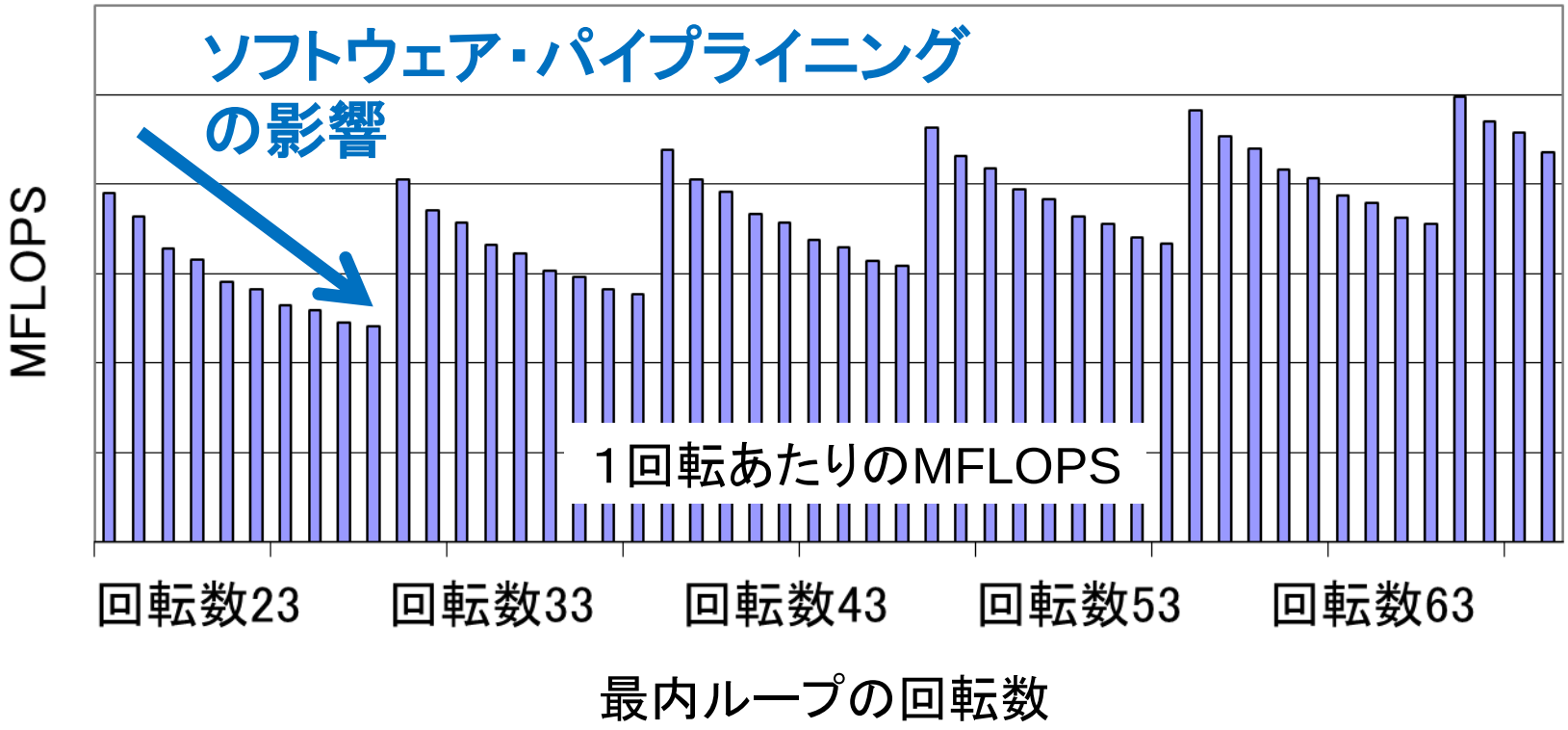
-Kfast,parallel,nosimd,nounroll,noprefetch

使いやすさ

FUJITSU

■ 回転数の影響を受ける最適化

１回転あたりのMFLOPSは、一定ではない



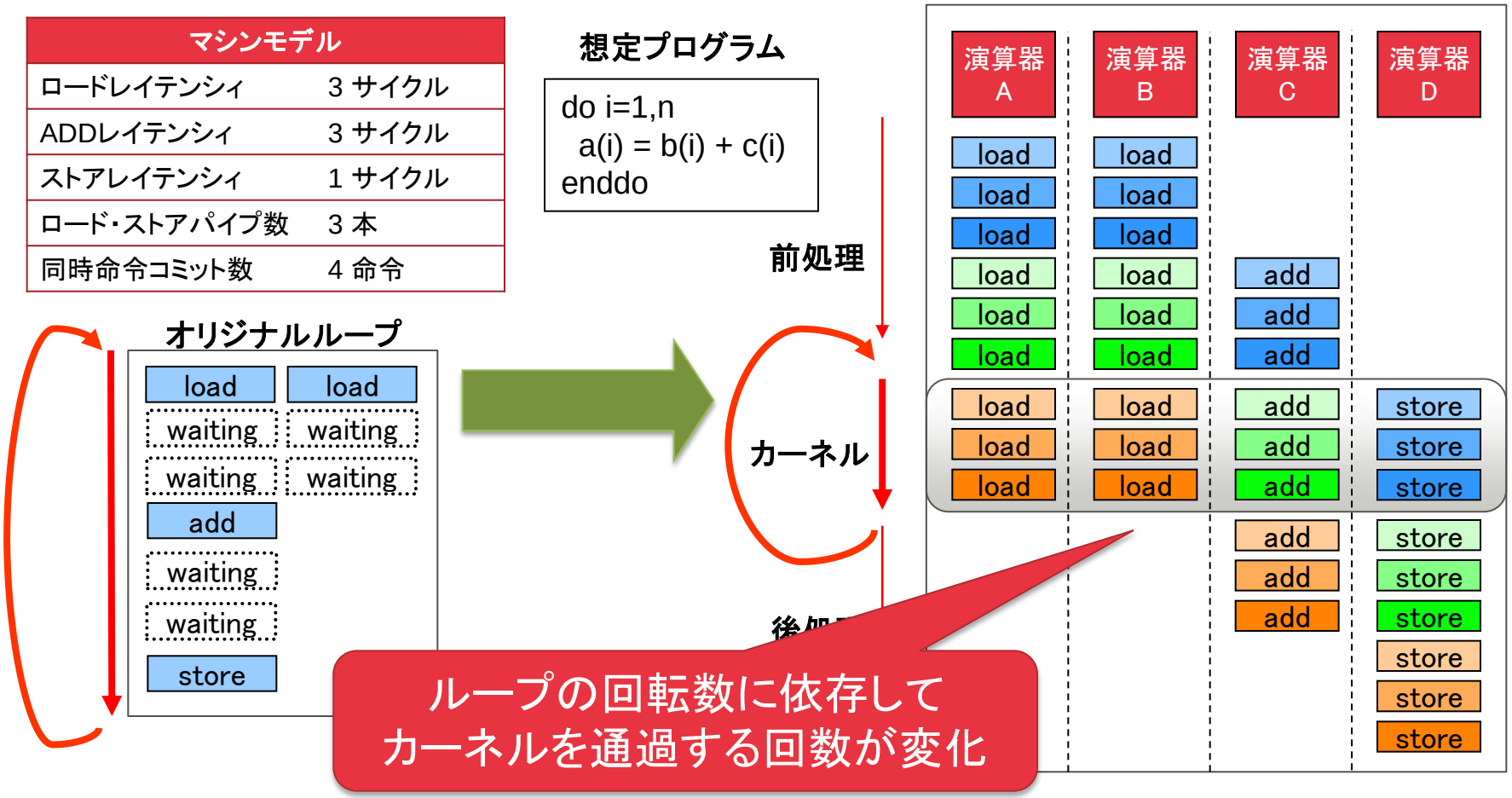
使いやすさ

■ ソフトウェア・パイプライニングの動き

マシンモデル	
ロードレイテンシィ	3 サイクル
ADDレイテンシィ	3 サイクル
ストアレイテンシィ	1 サイクル
ロード・ストアパイプ数	3 本
同時命令コミット数	4 命令

想定プログラム

```
do i=1,n
  a(i) = b(i) + c(i)
enddo
```



使いやすさ

FUJITSU

■ ポイント

- コーディング技法によって実行性能は大きく変化する
- 最適化によっても変化する
- 回転数など動作状況によっても変化する
- 課題はハードウェアだけではない

将来のハードウェアを考える

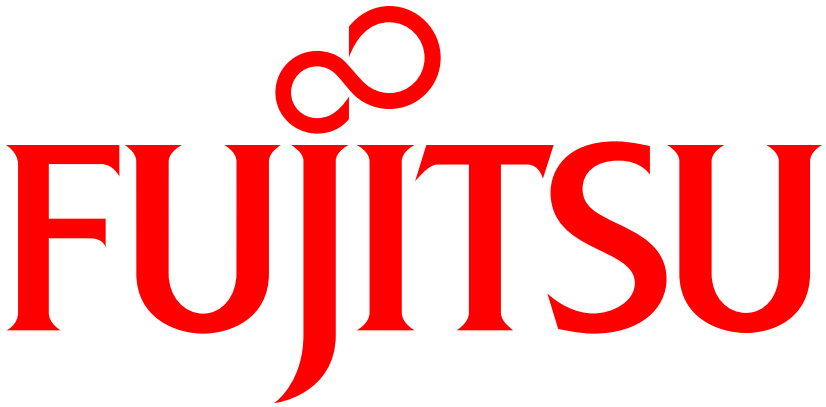
FUJITSU

■ まとめ

- **アプリケーション、コンパイラなどの努力も必要**
  ハードウェアを使いこなす、使い切る

- **変化に対する理由が明確なハードウェア/システム**
  メッセージ、数値化、視覚化、など

- **ヒューマンセントリック**
  人が理解できる



shaping tomorrow with you