

PSkel-MPPA: Uma Adaptação do Framework PSkel para o Processador Manycore MPPA-256

Emmanuel Podestá Junior

Orientação: Márcio B. Castro (UFSC)

Departamento de Informática e Estatística (INE)
Universidade Federal de Santa Catarina (UFSC)
emmanuel.podesta@grad.ufsc.br

21 Junho 2017

Contexto

Introdução

Contexto

Esqueletos Paralelos

Padrão Estêncil

Framework PSkel

MPPA-256

Proposta

Contribuições

Implementação

Resultados

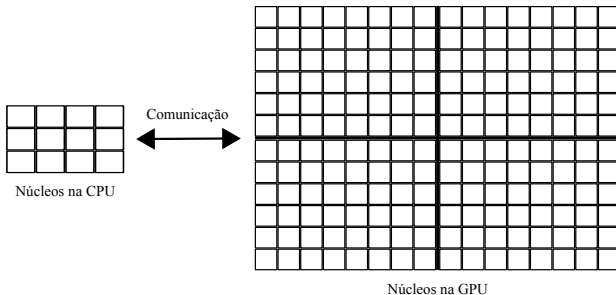
Conclusões e

Trabalhos

Futuros

Referências

- Aplicações complexas podem sobrecarregar a CPU
- Arquiteturas auxiliares
 - Ex.: Aceleradores
- **Computação de Alto Desempenho**
 - CPU e aceleradores (GPUs, manycores)
 - Maiores ganhos em desempenho



Introdução

Contexto

Esqueletos Paralelos

Padrão Estêncil

Framework PSkel

MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

Futuros

Referências

Dificuldades

- Programação Híbrida
- Comunicação entre CPUs e aceleradores
- Particionamento inteligente de tarefas

Esqueletos Paralelos

Introdução

Contexto

Esqueletos Paralelos

Padrão Estêncil

Framework PSkel

MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

Futuros

Referências

Existem vários padrões paralelos [McCool, 2010]

- Map, reduce, scan, **estêncil**, entre outros.
- Esqueletos abstraem os padrões paralelos
- Complexidade reduzida

***Frameworks* baseados no padrão estêncil:**

- SkelCL [Steuwer et al., 2011]
- SkePU [Enmyren and Kessler, 2010]
- PSkel [Pereira et al., 2015]

Padrão Estêncil

Introdução

Contexto

Esqueletos Paralelos

Padrão Estêncil

Framework PSkel

MPPA-256

Proposta

Contribuições

Implementação

Resultados

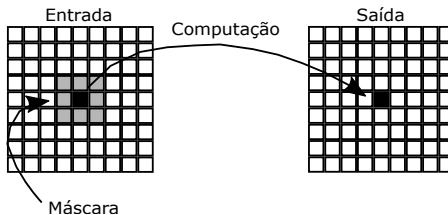
Conclusões e

Trabalhos

Futuros

Referências

- Cada célula da matriz é computada em função dos valores de seus vizinhos
- Computação Iterativa



Padrão Estêncil

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

Futuros

Referências

```
void jacobi(int tsteps, int N, float *A, float *B){  
    int t, i, j;  
    float c1 = 0.2;  
  
    for (t = 0; t < tsteps; t++){  
        for (i = 1; i < N-1; i++){  
            for (j = 1; j < N-1; j++){  
                B[i,j] = c1 * (A[i,j] + A[i,j-1] + A[i,j+1]  
                               + A[i+1,j] + A[i-1,j]);  
  
                for (i = 1; i < N-1; i++){  
                    for (j = 1; j < N-1; j++){  
                        A[i,j] = B[i,j]  
                    }  
                }  
            }  
        }  
    }  
}
```

Padrão Estêncil

Introdução

Contexto

Esqueletos Paralelos

Padrão Estêncil

Framework PSkel

MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

Futuros

Referências

```
void jacobi(int tsteps, int N, float *A, float *B){
    int t, i, j;
    float c1 = 0.2;

    for (t = 0; t < tsteps; t++){
        for (i = 1; i < N-1; i++){
            for (j = 1; j < N-1; j++){
                B[i,j] = c1 * (A[i,j] + A[i,j-1] + A[i,j+1]
                               + A[i+1,j] + A[i-1,j]);

            for (i = 1; i < N-1; i++){
                for (j = 1; j < N-1; j++){
                    A[i,j] = B[i,j]
                }
            }
        }
    }
```

Padrão Estêncil

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

Futuros

Referências

```
void jacobi(int tsteps, int N, float *A, float *B){  
    int t, i, j;  
    float c1 = 0.2;  
  
    for (t = 0; t < tsteps; t++){  
        for (i = 1; i < N-1; i++){  
            for (j = 1; j < N-1; j++){  
                B[i,j] = c1 * (A[i,j] + A[i,j-1] + A[i,j+1]  
                               + A[i+1,j] + A[i-1,j]);  
  
                for (i = 1; i < N-1; i++){  
                    for (j = 1; j < N-1; j++){  
                        A[i,j] = B[i,j]  
                    }  
                }  
            }  
        }  
    }  
}
```


Padrão Estêncil

Introdução

Contexto

Esqueletos Paralelos

Padrão Estêncil

Framework PSkel

MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

Futuros

Referências

```
void jacobi(int tsteps, int N, float *A, float *B){
    int t, i, j;
    float c1 = 0.2;

    for (t = 0; t < tsteps; t++){
        for (i = 1; i < N-1; i++)
            for (j = 1; j < N-1; j++)
                B[i,j] = c1 * (A[i,j] + A[i,j-1] + A[i,j+1]
                               + A[i+1,j] + A[i-1,j]);

        for (i = 1; i < N-1; i++)
            for (j = 1; j < N-1; j++)
                A[i,j] = B[i,j]
    }
}
```

Padrão Estêncil

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

Futuros

Referências

```
void jacobi(int tsteps, int N, float *A, float *B){  
    int t, i, j;  
    float c1 = 0.2;  
  
    for (t = 0; t < tsteps; t++){  
        for (i = 1; i < N-1; i++){  
            for (j = 1; j < N-1; j++){  
                B[i,j] = c1 * (A[i,j] + A[i,j-1] + A[i,j+1]  
                               + A[i+1,j] + A[i-1,j]);  
  
                for (i = 1; i < N-1; i++){  
                    for (j = 1; j < N-1; j++){  
                        A[i,j] = B[i,j]  
                    }  
                }  
            }  
        }  
    }  
}
```

Padrão Estêncil

Introdução

Contexto

Esqueletos Paralelos

Padrão Estêncil

Framework PSkel

MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

Futuros

Referências

```
void jacobi(int tsteps, int N, float *A, float *B){
    int t, i, j;
    float c1 = 0.2;

    for (t = 0; t < tsteps; t++){
        for (i = 1; i < N-1; i++){
            for (j = 1; j < N-1; j++){
                B[i,j] = c1 * (A[i,j] + A[i,j-1] + A[i,j+1]
                               + A[i+1,j] + A[i-1,j]);

            for (i = 1; i < N-1; i++){
                for (j = 1; j < N-1; j++){
                    A[i,j] = B[i,j]
                }
            }
        }
    }
```

Padrão Estêncil

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

Futuros

Referências

```
void jacobi(int tsteps, int N, float *A, float *B){  
    int t, i, j;  
    float c1 = 0.2;  
  
    for (t = 0; t < tsteps; t++){  
        for (i = 1; i < N-1; i++){  
            for (j = 1; j < N-1; j++){  
                B[i,j] = c1 * (A[i,j] + A[i,j-1] + A[i,j+1]  
                               + A[i+1,j] + A[i-1,j]);  
  
                for (i = 1; i < N-1; i++){  
                    for (j = 1; j < N-1; j++){  
                        A[i,j] = B[i,j]  
                    }  
                }  
            }  
        }  
    }  
}
```

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências

Objetivo

- Oferecer suporte para execução paralela de aplicações do padrão estêncil em ambientes heterogêneos (CPU e GPU) [Pereira et al., 2015]

Funcionamento

- O usuário descreve o *kernel* principal da computação estêncil
- O framework se encarrega de distribuir a computação na CPU e GPU
- Transferências de dados e particionamento de tarefas de maneira transparente

Framework PSkel

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências

```
__parallel__ void
stencilKernel(Array2D<float> A, Array2D<float> B,
Mask2D<int> mask, struct Arguments args,
int x, int y){
    B(x,y) = args.alpha * (A(x,y+1) + A(x,y-1) +
        A(x+1,y) + A(x-1,y));
}

void main(){
    /* ... */
    Array2D<float> input(A,M,N);
    Array2D<float> output(B,M,N);
    int neighbors = {{0,1}, {-1,0}, {1,0}, {-1,0}};
    Mask2D<int> mask(4, neighbors);
    struct Arguments args(alpha, beta);
    /* ... */
    Stencil2D<Array2D<float>, Mask2D<int>, Arguments>
    jacobi(A,B,args);
    jacobi.runIterative(device::GPU, timesteps, 1.0);
    /* ... */
}
```

Framework PSkel

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

Futuros

Referências

```
__parallel__ void
stencilKernel(Array2D<float> A, Array2D<float> B,
Mask2D<int> mask, struct Arguments args,
int x, int y){
    B(x,y) = args.alpha * (A(x,y+1) + A(x,y-1) +
        A(x+1,y) + A(x-1,y));
}

void main(){
    /* ... */
    Array2D<float> input(A,M,N);
    Array2D<float> output(B,M,N);
    int neighbors = {{0,1}, {-1,0}, {1,0}, {-1,0}};
    Mask2D<int> mask(4, neighbors);
    struct Arguments args(alpha, beta);
    /* ... */
    Stencil2D<Array2D<float>, Mask2D<int>, Arguments>
    jacobi(A,B,args);
    jacobi.runIterative(device::GPU, timesteps, 1.0);
    /* ... */
}
```

Framework PSkel

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

Futuros

Referências

```
--parallel__ void
stencilKernel(Array2D<float> A, Array2D<float> B,
Mask2D<int> mask, struct Arguments args,
int x, int y){
    B(x,y) = args.alpha * (A(x,y+1) + A(x,y-1) +
        A(x+1,y) + A(x-1,y));
}

void main(){
    /* ... */
    Array2D<float> input(A,M,N);
    Array2D<float> output(B,M,N);
    int neighbors = {{0,1}, {-1,0}, {1,0}, {-1,0}};
    Mask2D<int> mask(4, neighbors);
    struct Arguments args(alpha, beta);
    /* ... */
    Stencil2D<Array2D<float>, Mask2D<int>, Arguments>
    jacobi(A,B,args);
    jacobi.runIterative(device::GPU, timesteps, 1.0);
    /* ... */
}
```


Framework PSkel

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

Futuros

Referências

```
--parallel__ void
stencilKernel(Array2D<float> A, Array2D<float> B,
Mask2D<int> mask, struct Arguments args,
int x, int y){
    B(x,y) = args.alpha * (A(x,y+1) + A(x,y-1) +
        A(x+1,y) + A(x-1,y));
}

void main(){
    /* ... */
    Array2D<float> input(A,M,N);
    Array2D<float> output(B,M,N);
    int neighbors = {{0,1}, {-1,0}, {1,0}, {-1,0}};
    Mask2D<int> mask(4, neighbors);
    struct Arguments args(alpha, beta);
    /* ... */
    Stencil2D<Array2D<float>, Mask2D<int>, Arguments>
    jacobi(A,B,args);
    jacobi.runIterative(device::GPU, timesteps, 1.0);
    /* ... */
}
```

Framework PSkel

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

Futuros

Referências

```
--parallel__ void
stencilKernel(Array2D<float> A, Array2D<float> B,
Mask2D<int> mask, struct Arguments args,
int x, int y){
    B(x,y) = args.alpha * (A(x,y+1) + A(x,y-1) +
        A(x+1,y) + A(x-1,y));
}

void main(){
    /* ... */
    Array2D<float> input(A,M,N);
    Array2D<float> output(B,M,N);
    int neighbors = {{0,1}, {-1,0}, {1,0}, {-1,0}};
    Mask2D<int> mask(4, neighbors);
    struct Arguments args(alpha, beta);
    /* ... */
    Stencil2D<Array2D<float>, Mask2D<int>, Arguments>
    jacobi(A,B,args);
    jacobi.runIterative(device::GPU, timesteps, 1.0);
    /* ... */
}
```

Framework PSkel

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

Futuros

Referências

```
--parallel__ void
stencilKernel(Array2D<float> A, Array2D<float> B,
Mask2D<int> mask, struct Arguments args,
int x, int y){
    B(x,y) = args.alpha * (A(x,y+1) + A(x,y-1) +
        A(x+1,y) + A(x-1,y));
}

void main(){
    /* ... */
    Array2D<float> input(A,M,N);
    Array2D<float> output(B,M,N);
    int neighbors = {{0,1}, {-1,0}, {1,0}, {-1,0}};
    Mask2D<int> mask(4, neighbors);
    struct Arguments args(alpha, beta);
    /* ... */
    Stencil2D<Array2D<float>, Mask2D<int>, Arguments>
    jacobi(A,B,args);
    jacobi.runIterative(device::GPU, timesteps, 1.0);
    /* ... */
}
```

Framework PSkel

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

Futuros

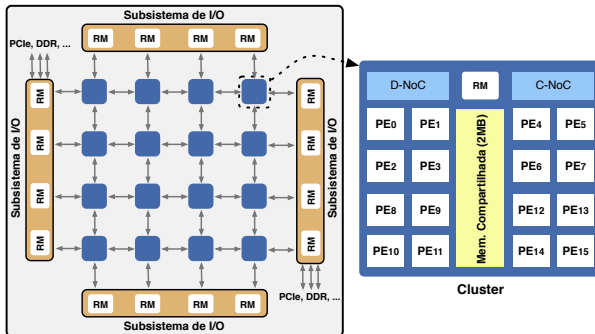
Referências

```
__parallel__ void
stencilKernel(Array2D<float> A, Array2D<float> B,
Mask2D<int> mask, struct Arguments args,
int x, int y){
    B(x,y) = args.alpha * (A(x,y+1) + A(x,y-1) +
        A(x+1,y) + A(x-1,y));
}

void main(){
    /* ... */
    Array2D<float> input(A,M,N);
    Array2D<float> output(B,M,N);
    int neighbors = {{0,1}, {-1,0}, {1,0}, {-1,0}};
    Mask2D<int> mask(4, neighbors);
    struct Arguments args(alpha, beta);
    /* ... */
    Stencil2D<Array2D<float>, Mask2D<int>, Arguments>
    jacobi(A,B,args);
    jacobi.runIterative(device::GPU, timesteps, 1.0);
    /* ... */
}
```

Características

- Manycore de baixo consumo de energia
- 256 núcleos organizados em 16 clusters
- 4 subsistemas de E/S
- Comunicação feita por NoC *torus* 2D



Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

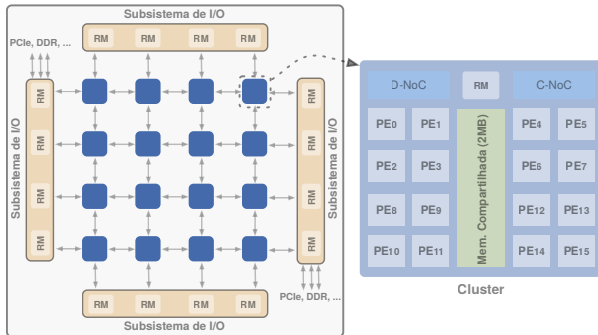
Resultados

Conclusões e Trabalhos Futuros

Referências

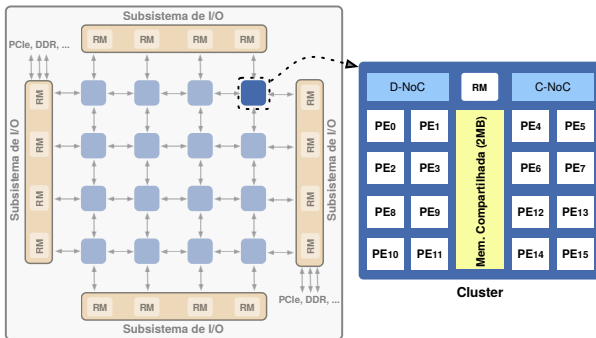
Características

- Manycore de baixo consumo de energia
- 256 núcleos organizados em 16 clusters
- 4 subsistemas de E/S
- Comunicação feita por NoC *torus* 2D



Características

- Manycore de baixo consumo de energia
- 256 núcleos organizados em 16 clusters
- 4 subsistemas de E/S
- Comunicação feita por NoC *torus* 2D



Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

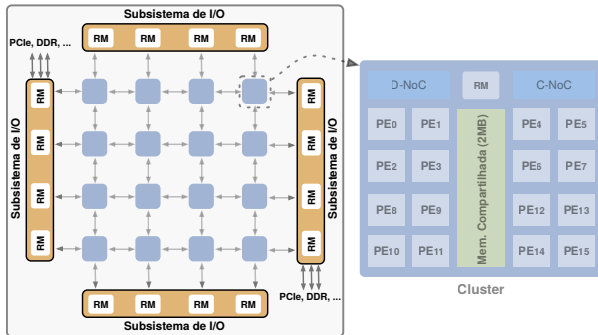
Resultados

Conclusões e Trabalhos Futuros

Referências

Características

- Manycore de baixo consumo de energia
- 256 núcleos organizados em 16 clusters
- 4 subsistemas de E/S
- Comunicação feita por NoC *torus* 2D



Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

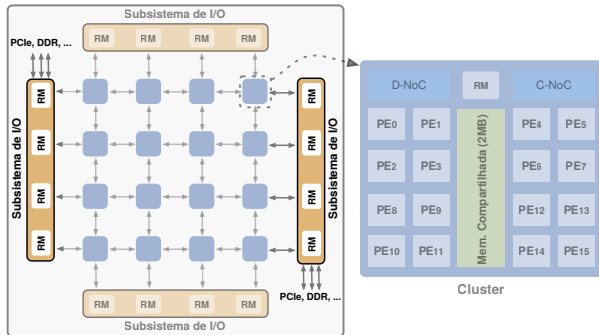
Resultados

Conclusões e Trabalhos Futuros

Referências

Características

- Manycore de baixo consumo de energia
- 256 núcleos organizados em 16 clusters
- 4 subsistemas de E/S
- Comunicação feita por NoC *torus* 2D



Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

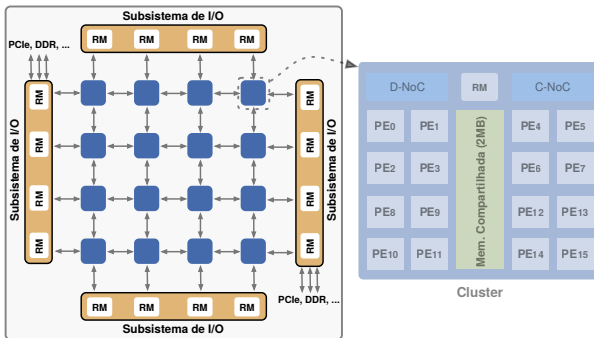
Resultados

Conclusões e Trabalhos Futuros

Referências

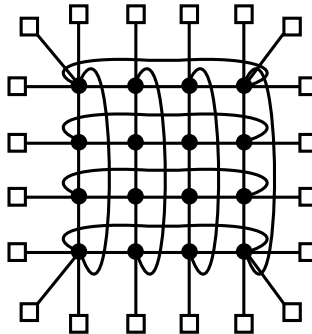
Características

- Manycore de baixo consumo de energia
- 256 núcleos organizados em 16 clusters
- 4 subsistemas de E/S
- Comunicação feita por NoC *torus* 2D



Características

- Manycore de baixo consumo de energia
- 256 núcleos organizados em 16 clusters
- 4 subsistemas de E/S
- Comunicação feita por NoC *torus* 2D



Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências

Introdução

Contexto

Esqueletos Paralelos

Padrão Estêncil

Framework PSkel

MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

Futuros

Referências

Dificuldades encontradas no desenvolvimento de aplicações para o MPPA-256

- **Comunicação via NoC:** toda a comunicação é feita através de uma API proprietária de baixo nível similar a *POSIX Interprocess Communication (IPC)*
- **Memória limitada nos *clusters*:** cada *cluster* possui somente uma memória local de 2MB
- **Baixo nível de abstração:** toda a comunicação é gerenciada pelo desenvolvedor de maneira explícita

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências

Objetivo Geral

- Adaptar o *Framework* PSkel para o processador MPPA-256
- Simplificar o desenvolvimento para o MPPA-256
- Desenvolvedor poderá aproveitar os benefícios do processador
- Aplicações desenvolvidas poderão ser portadas sem alteração

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências

Objetivos Específicos

- Definir uma estratégia de distribuição de dados entre os *clusters* do MPPA-256
- Propor e implementar técnicas que permitam reduzir os custos de comunicação na NoC
- Adaptar as principais classes e abstrações existentes no PSkel para o processador MPPA-256
- Realizar uma análise de desempenho e energia sobre a solução proposta
- Realizar comparações de desempenho e consumo de energia com um processador *multicore* atual

Contribuições

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências

- PODESTA JUNIOR, E. ; MARQUES, B. ; CASTRO, M. **Energy Efficient Stencil Computations on the Low-Power Manycore MPPA-256 Processor**. In: Conferência Européia Internacional de Computação Paralela e Distribuída (EURO-PAR), 2018, Turin, Itália.
- PODESTA JUNIOR, E. ; PEREIRA, A. D. ; ROCHA, R. C. O. ; CASTRO, MÁRCIO ; GOES, L. F. W. **Execução Energeticamente Eficiente de Aplicações Estêncil com o Processador Manycore MPPA-256**. In: Simpósio em Sistemas Computacionais de Alto Desempenho (WSCAD), 2017, Campinas, São Paulo.
- PODESTA JUNIOR, E. ; PEREIRA, A. D. ; ROCHA, R. C. O. ; CASTRO, M. ; GOES, L. F. W. **Uma Implementação do Framework PSkel com Suporte a Aplicações Estêncil Iterativas para o Processador MPPA-256**. In: Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul (ERAD/RS), 2017, Ijuí, Rio Grande do Sul.
- PODESTA JUNIOR, E. ; PEREIRA, A. D. ; PENNA, P. H. ; ROCHA, R. C. O. ; CASTRO, M. ; GOES, L. F. W. **PSkel-MPPA: Uma Adaptação do Framework PSkel para o Processador Manycore MPPA-256**. In: Escola Regional de Alto Desempenho do Estado do Rio Grande do Sul (ERAD/RS), 2016, São Leopoldo, Rio Grande do Sul.

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências

- Modelo mestre/trabalhador
- **Mestre (subsistema de I/O)**
 - Subdivide a matriz de entrada em *tiles* e os envia sob demanda aos trabalhadores (*clusters*)
 - Envio feito de acordo com o modelo *Round-Robin*
 - Particionamento flexível dos dados entre os *clusters* (*tiles* de dimensão arbitrária)

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências

Trabalhador (*cluster*)

- Recebe o *tile*, realiza a computação do *kernel* estêncil sobre o *tile* e envia a resposta ao mestre
- Computação realizada por meio de diretivas OpenMP
- Repete a computação para cada *tile* atribuído pelo mestre

Problema

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências

- No padrão estêncil temos computações iterativas
- Grande número de comunicações entre o processo mestre e trabalhador
- Atribuir parte das iterações para o processo trabalhador
- Redução no número de comunicações

Tiling Trapezoidal

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

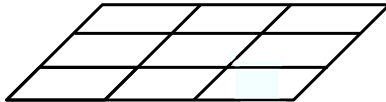
Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências



Tiling Trapezoidal

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

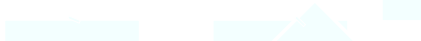
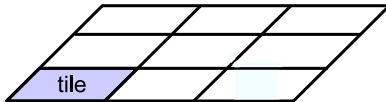
Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências



Tiling Trapezoidal

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

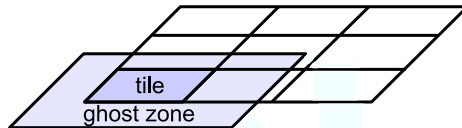
Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências



Tiling Trapezoidal

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

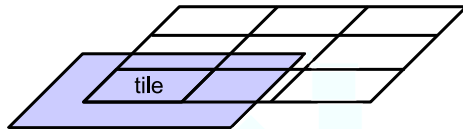
Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências



Tiling Trapezoidal

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

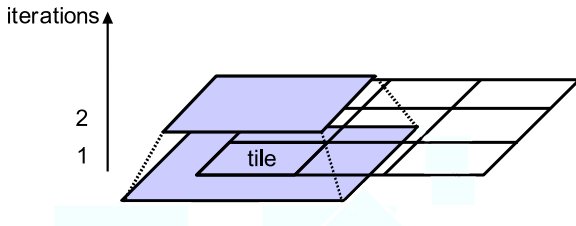
Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências



Tiling Trapezoidal

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

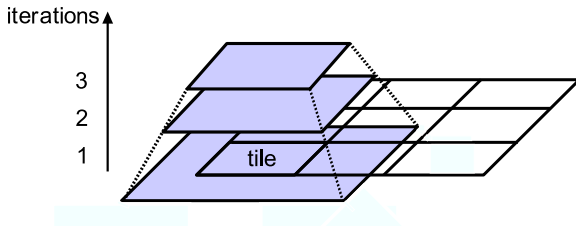
Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências



Tiling Trapezoidal

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

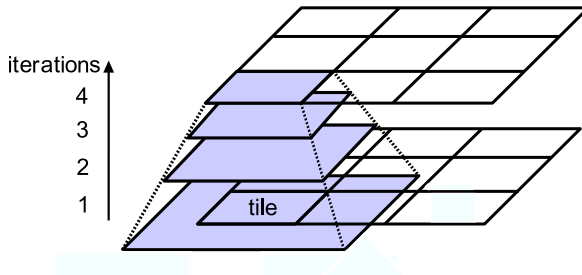
Resultados

Conclusões e

Trabalhos

Futuros

Referências



Tiling Trapezoidal

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

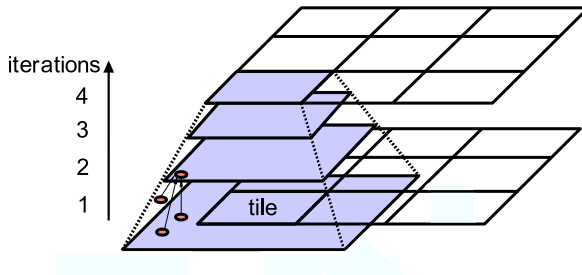
Resultados

Conclusões e

Trabalhos

Futuros

Referências



Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

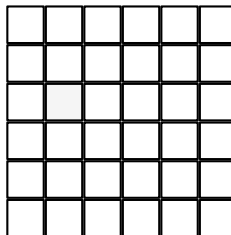
Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências



Matriz de entrada bidimensional
Processo Mestre no Subsistema E/S

Comunicação

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

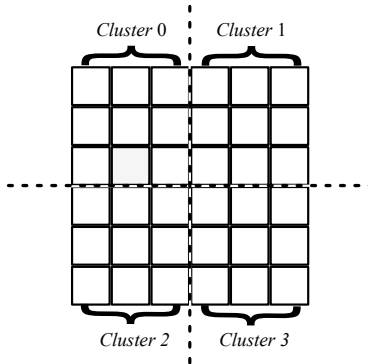
Resultados

Conclusões e

Trabalhos

Futuros

Referências



Matriz de entrada bidimensional
Processo Mestre no Subsistema E/S

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

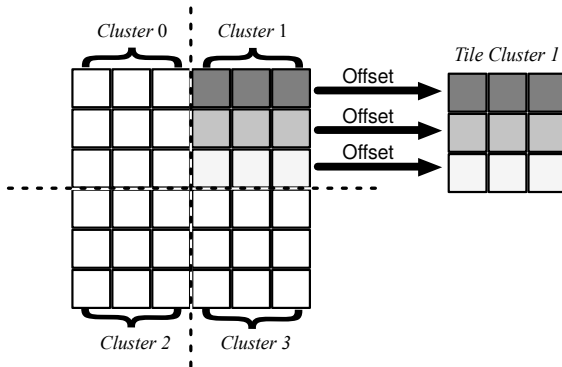
Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências

Comunicação Mestre-Trabalhador



Matriz de entrada bidimensional
Processo Mestre no Subsistema E/S

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

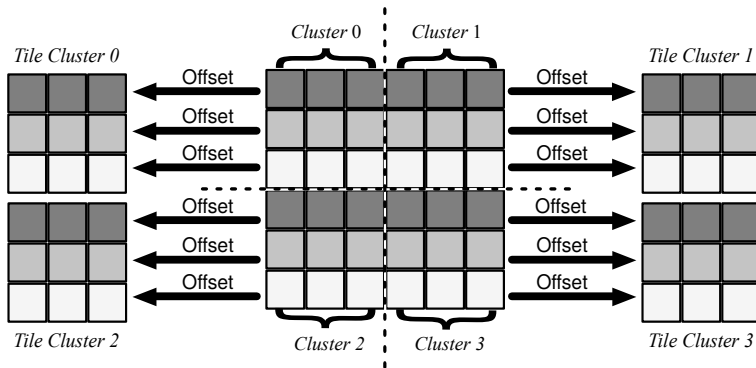
Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências

Comunicação Mestre-Trabalhador



Matriz de entrada bidimensional
Processo Mestre no Subsistema E/S

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

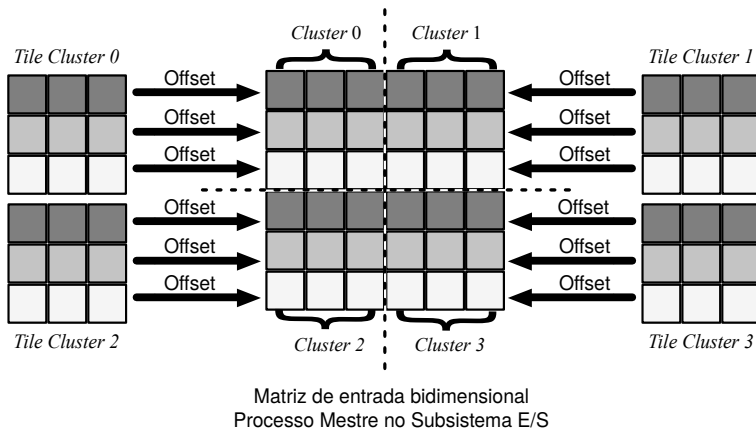
Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências

Comunicação Trabalhador-Mestre



Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e
Trabalhos
Futuros

Referências

■ **Processador *manycore* MPPA-256**

- Calculado a média do tempo e energia gastos sobre 5 execuções
- Baixa variabilidade entre execuções

■ **Processador *multicore* Intel Xeon E5-2640 v4**

- Calculado a média do tempo e energia gastos sobre 30 execuções
- Desvio padrão menor que 1%
- Todos os experimentos utilizaram 30 iterações

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências

■ Aplicações

- **Fur:** simulação de padrões de pigmento em pelos de animais [Podestá Jr. et al., 2016]
- **GoL:** autômato celular que implementa o Jogo da Vida de Conway [Pereira et al., 2015]
- **Jacobi:** método iterativo de Jacobi para a resolução de equações matriciais [Pereira et al., 2015]

Introdução

Contexto

Esqueletos Paralelos

Padrão Estêncil

Framework PSkel

MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

Trabalhos

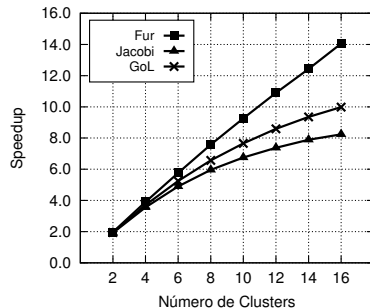
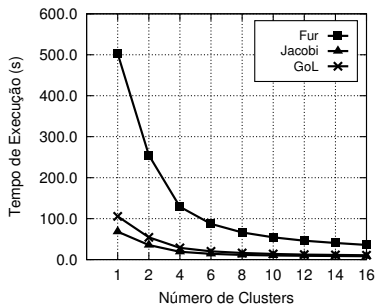
Futuros

Referências

Teste de Escalabilidade

■ Matriz: 2048x2048

■ Tile: 128x128



Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

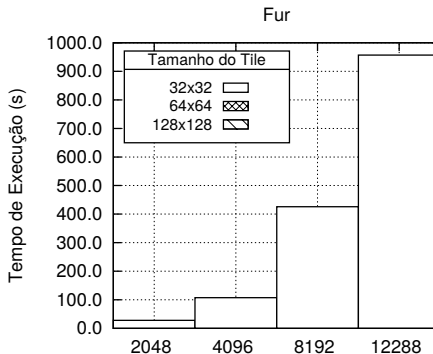
Trabalhos

Futuros

Referências

Tamanho dos *tiles* vs. Desempenho

■ 16 *clusters* utilizados



Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

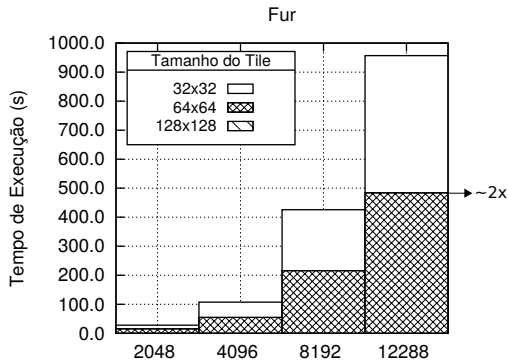
Trabalhos

Futuros

Referências

Tamanho dos *tiles* vs. Desempenho

■ 16 *clusters* utilizados



Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e

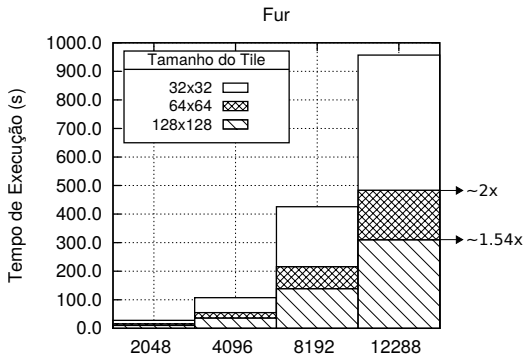
Trabalhos

Futuros

Referências

Tamanho dos *tiles* vs. Desempenho

■ 16 *clusters* utilizados



Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

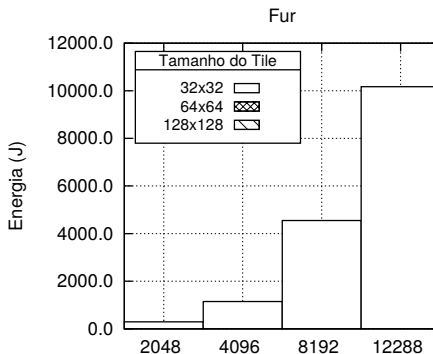
Resultados

Conclusões e Trabalhos Futuros

Referências

Tamanho dos *tiles* vs. Energia

■ 16 *clusters* utilizados



Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

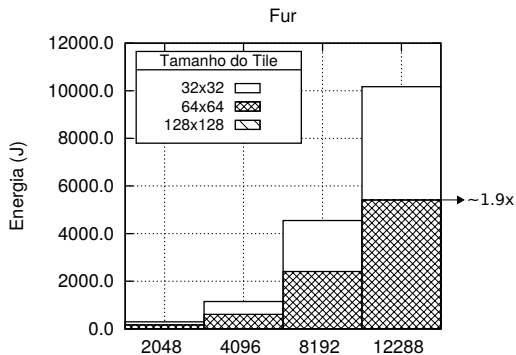
Resultados

Conclusões e Trabalhos Futuros

Referências

Tamanho dos *tiles* vs. Energia

■ 16 *clusters* utilizados



Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

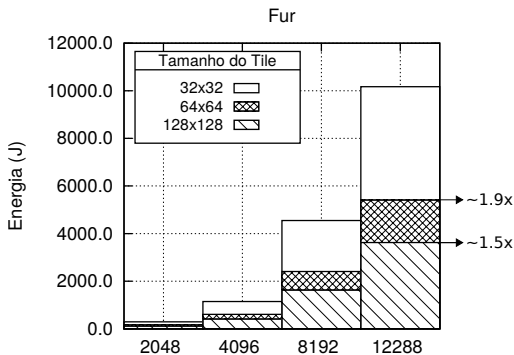
Resultados

Conclusões e Trabalhos Futuros

Referências

Tamanho dos *tiles* vs. Energia

■ 16 *clusters* utilizados



Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e
Trabalhos
Futuros

Referências

MPPA vs. Intel Xeon

- Matriz: 12288x12288
- *Tile*: 128x128
- 16 *clusters* utilizados no MPPA
- 10 *threads* sem *hyperthreading* utilizadas no Intel Xeon

MPPA vs. Intel Xeon

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

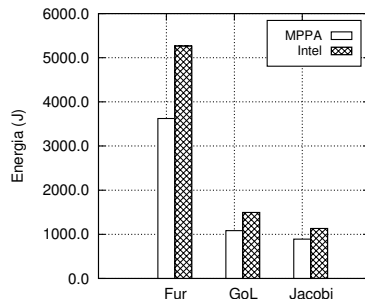
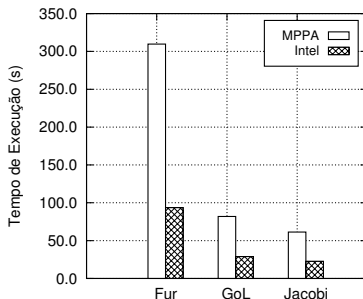
Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências



- Para as aplicações Fur, GoL e Jacobi:
 - O Intel Xeon é 3.30x, 2.83x e 2.69x, respectivamente, mais rápido
 - O MPPA consome 1.45x, 1.38x e 1.27x, respectivamente, menos energia

Conclusões

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados


Conclusões e Trabalhos Futuros

Referências

- A adaptação apresenta uma eficiência energética superior ao Intel Xeon
- A adaptação demonstrou um bom potencial
- A comunicação tem impacto sobre o tempo e energia obtidos
- Boa escalabilidade em relação à variação dos *clusters* e variação do tamanho dos *tiles*

Trabalhos Futuros

- Realizar experimentos com estruturas tridimensionais
- Reduzir os sobrecustos de comunicação
- Comparar a adaptação com outros processadores embarcados



PSkel-MPPA: Uma Adaptação do Framework PSkel para o Processador Manycore MPPA-256

Emmanuel Podestá Junior

Orientação: Márcio B. Castro (UFSC)

Departamento de Informática e Estatística (INE)
Universidade Federal de Santa Catarina (UFSC)
emmanuel.podesta@grad.ufsc.br

21 Junho 2017

Referências

Introdução

Contexto
Esqueletos Paralelos
Padrão Estêncil
Framework PSkel
MPPA-256

Proposta

Contribuições

Implementação

Resultados

Conclusões e Trabalhos Futuros

Referências



Enmyren, J. and Kessler, C. W. (2010).

SkePU: A Multi-backend Skeleton Programming Library for multi-GPU Systems.

In *International Workshop on High-level Parallel Programming and Applications (HLPP)*, pages 5–14, Baltimore, USA. ACM.



McCool, M. D. (2010).

Structured parallel programming with deterministic patterns.

In *Proceedings of the 2Nd USENIX Conference on Hot Topics in Parallelism, HotPar'10*, pages 5–5, Berkeley, CA, USA. USENIX Association.



Pereira, A. D., Ramos, L., and Góes, L. F. W. (2015).

PSkel: A Stencil Programming Framework for CPU-GPU Systems.

CCPE, 27(17):4938–4953.



Podestá Jr., E., Pereira, A. D., Penna, P. H., Rocha, R. C., Castro, M., and Góes, L. F. W. (2016).

PSkel-MPPA: Uma Adaptação do Framework PSkel para o Processador Manycore MPPA-256.

In *ERAD/RS*, pages 299–302, São Leopoldo, Brazil. SBC.



Steuwer, M., Kegel, P., and Gorlatch, S. (2011).

SkelCL: A Portable Skeleton Library for High-Level GPU Programming.

In *IEEE International Symposium on Parallel and Distributed Processing Workshops (IPDPSW)*, pages 1176–1182, Shanghai, China. IEEE Computer Society.