

MC970/MO644

Programação Paralela na Nuvem usando OpenMP

Hervé Yviquel, Guido Araújo

`herve.yviquel@ic.unicamp.br`

O datacenter como Supercomputador



Massivo Poder de Processamento Paralelo

Arquitetura distribuída dos datacenters

- Até 15 milhões de núcleos ??
 - Até 80.000 servidores
 - Até 8 processadores por placa mãe
 - Até 24 núcleos por processador

Nova possibilidade de aplicações paralelas

- Simulação do Bóson de Higgs com +/- 58.000 núcleos
- Renderização do filme *Monster Inc.* de Pixar com 24.000 núcleos (29 horas por frame => 100 milhões de horas)

DID YOU MOVE ALL THE SERVERS TO "THE CLOUD" YET?

KIP

GARY

EXIT

JUST FINISHED PACKING THEM UP...NOW WHAT IS THE ADDRESS TO "THE CLOUD" AGAIN?

SERVERS

SERVERS

SERVERS

SERVERS

SERVERS

CORDS

CORDS

CORDS

CORDS

ALBER

© 2011 Diane Alber

Cadê os Nuvens ?

Fornecedores de nuvem publica

Serviço comercial (Amazon Web Service, Microsoft Azure, Google Cloud, etc.)

Nuvens privadas

Dedicado para uma empresa (Bancos, etc)



É basicamente um supercomputador barato !!

Especificidade do Nuvem

Técnica de virtualização

“Separar sistema operacional e componentes físicos”

Máquina Virtual (e.g. Xen or KVM)

Sistema Operacional da Nuvem (e.g. OpenStack)

Flexibilidade do uso

Compartilhar os recursos

Disponibilidade e recuperação

Isolamento e segurança

Pay-as-you-go – “Paga apenas pelos serviços consumidos”

Domínios de Aplicação

Solução para a ascensão do “Big Data”

- Redes sociais (Facebook, Twitter)
- Multimídia (Netflix, Spotify)

Mas pode ser útil para outros

- Aplicações científicas (HPC)
- Aplicações mobile (Mobile cloud offloading)
- Internet das Coisas (IoT)

**Mas como programar a arquitetura distribuída
do nuvem ??**

Programar Aplicação Paralela no Nuvem?

Message Passing Interface (MPI)

Muito eficiente

Programação de baixo nível

Sem tolerância a falhas

Map-Reduce (e Spark)

Alta escalabilidade

Programação de alto nível

Tolerância a falhas

HPC

Big-data

Há ainda um problema

Não sou especialista de programação!



Escreve programas paralelos é difícil

- Não é tão natural



Integrar seu programa com a nuvem também

- Execução híbrida
- Varias linguagens de programação

Vamos simplificar !

(Re)introdução ao OpenMP

API para desenvolvimento de aplicativos paralelos

- Programação baseada em diretivas
 - Feito para ser simples e não precisa reescrever o código
- Suponha arquitetura de memória compartilhada

```
int MatMul(float *A, float *B, float *C) {  
    #pragma omp parallel for  
    for(int i=0; i < N; ++i)  
        for(int j = 0; j < N; ++j)  
            C[i * N + j] = 0;  
        for(int k = 0; k < N; ++k)  
            C[i * N + j] += A[i * N + k] * B[k * N + j];  
    return 0;  
}
```

Modelo do acelerador OpenMP

Extensão para programar aceleradores (v4.0+)

- Novas diretivas OpenMP
- Projetado para aceleradores locais (e.g. GPU)
- Modelo de arquitetura de “host-target”

```
int MatMul(float *A, float *B, float *C) {  
    #pragma omp target device(GPU)  
    #pragma omp map(to: A[:N*N], B[:N*N]) map(from: C[:N*N])  
    #pragma omp parallel for  
    for(int i=0; i < N; ++i)  
        for(int j = 0; j < N; ++j)  
            C[i * N + j] = 0;  
        for(int k = 0; k < N; ++k)  
            C[i * N + j] += A[i * N + k] * B[k * N + j];  
    return 0;  
}
```

A nuvem como um Acelerador

Vamos ser corajosos!!

- Introduzir a nuvem como um acelerador OpenMP
- Apenas outro acelerador disponível no seu computador

```
int MatMul(float *A, float *B, float *C) {  
    #pragma omp target device(CLOUD) ←  
    #pragma omp map(to: A[:N*N], B[:N*N]) map(from: C[:N*N])  
    #pragma omp parallel for  
    for(int i=0; i < N; ++i)  
        for(int j = 0; j < N; ++j)  
            C[i * N + j] = 0;  
        for(int k = 0; k < N; ++k)  
            C[i * N + j] += A[i * N + k] * B[k * N + j];  
    return 0;  
}
```

OpenMP + Nuvem = OmpCloud

Ambiente de descarregamento no nuvem

- Projeto de pesquisa
- Open-source (disponível no Github)
- Versão personalizada do compilador Clang / LLVM
- Descarregamento usando a biblioteca de OpenMP
- Computação em nuvem usando Spark

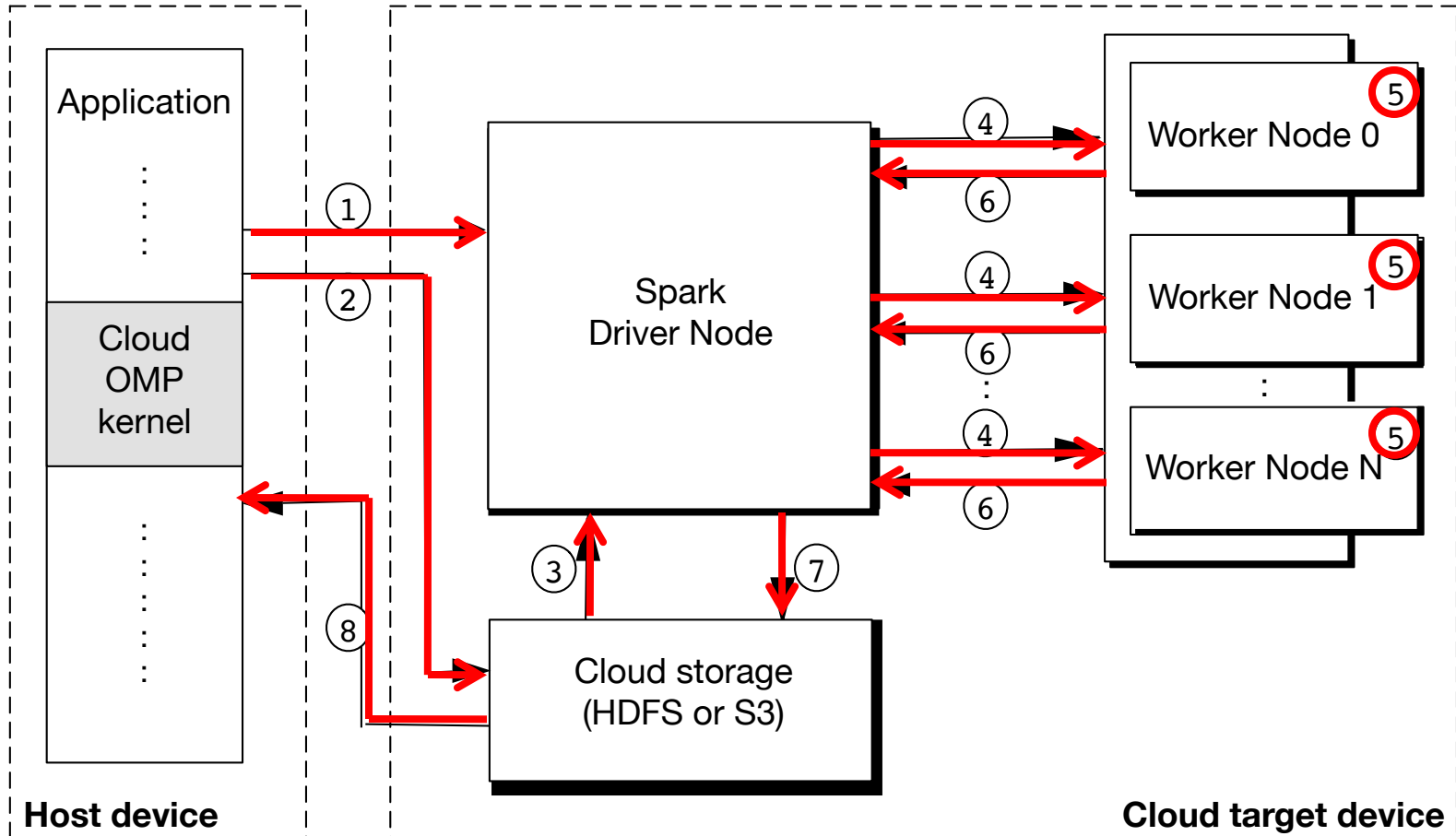
Consulte nosso site em ompcloud.org



Desenvolvimento com OmpCloud

1. Descrever o aplicativo usando o model de acelerador OpenMP
`#pragma omp target device(CLOUD)`
2. Compile com nossa versão personalizada de Clang/LLVM para gerar binarios
`clang -fopenmp -omptarget=linux-unknown-spark`
3. Instanciar um cluster do Spark usando o serviço em nuvem escolhido
(por ex. Microsoft Azure)
4. Configurar o runtime de OmpCloud com os credenciais para acessar a infraestrutura de nuvem
5. Execute o aplicativo!

Fluxo de Descarregamento



Implementação modular do modelo de aceleradores

Código do Host

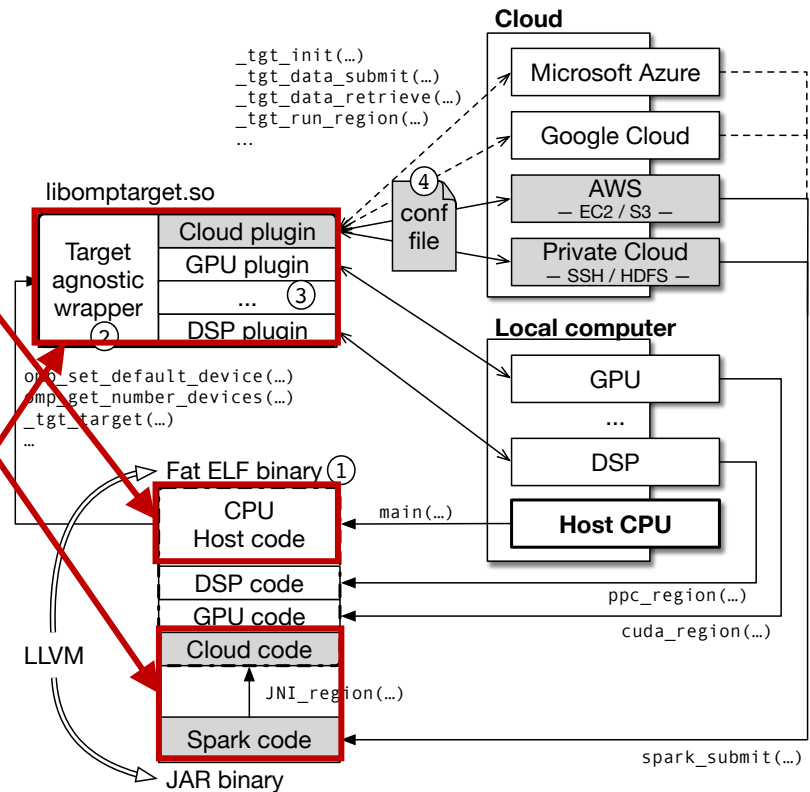
- Programa principal
- Chama funções para descarregar

Código do Acelerador

- Computação
- Código do Spark (em Scala)

Libreria de descarregamento

- Faz interações com o fornecedor de nuvem
- Configurável para cada fornecedor de nuvem



Configurar OmpCloud para um fornecedor de nuvem

- **Spark cluster**
 - URL e porta do nó *driver*
 - Configuração do Spark cluster
- **Armazenamento do nuvem**
 - HDFS server (URL, porta, usuario)
 - Amazon S3 (Access and secret keys, bucket, ...)
 - Pasta do trabalho
- **Runtime configuration**
 - Compression

Particionamento de Dados

É essencial em sistemas distribuídos

Por quê ?

- Mapear o bloco de dados para o nó do cluster usando-o
- Reduzir a sobrecarga de comunicação

Mas ...

- Não é possível determinar estaticamente em geral
- OpenMP não fornecem mecanismo para descrevê-lo

Estendemos OpenMP para conseguir expressa-lo


Extensão de OpenMP para Particionamento de Dados

```
int MatMul(float *A, float *B, float *C) {  
    #pragma omp target device(CLOUD)  
    #pragma omp map(to: A[:N*N], B[:N*N]) map(from: C[:N*N])  
    #pragma omp parallel for  
    for(int i=0; i < N; ++i)  
    → #pragma omp data map(to: A[i*N:(i+1)*N]) map(from: C[i*N:(i+1)*N])  
        for(int j = 0; j < N; ++j)  
            C[i * N + j] = 0;  
            for(int k = 0; k < N; ++k)  
                C[i * N + j] += A[i * N + k] * B[k * N + j];  
    return 0;  
}
```

Descrever a partição com a cláusulas *data map*

- Limites relacionados com o índice “i” do laço paralelo
- Já existe no padrão para descrever dados descarregados em várias regiões descarregadas
 - Dá semântica a um comportamento indefinido

Extensão de OpenMP para Particionamento de Dados

```
int MatMul(float *A, float *B, float *C) {  
    #pragma omp target device(CLOUD)  
    #pragma omp map(to: A[:N*N], B[:N*N]) map(from: C[:N*N])  
    #pragma omp parallel for  
    for(int i=0; i < N; ++i)  
     #pragma omp data map(to: A[i*N:(i+1)*N]) map(from: C[i*N:(i+1)*N])  
        for(int j = 0; j < N; ++j)  
            C[i * N + j] = 0;  
            for(int k = 0; k < N; ++k)  
                C[i * N + j] += A[i * N + k] * B[k * N + j];  
    return 0;  
}
```

Observe que B não está particionado aqui

- As matrizes são linearizadas na memória
- O particionamento B dependeria de "k" indexando as colunas
- Exigiria quebrar o cálculo
- Spark reduz a sobrecarga de transmissão usando BitTorrent

Corresponde ao Modelo de Execução do Spark

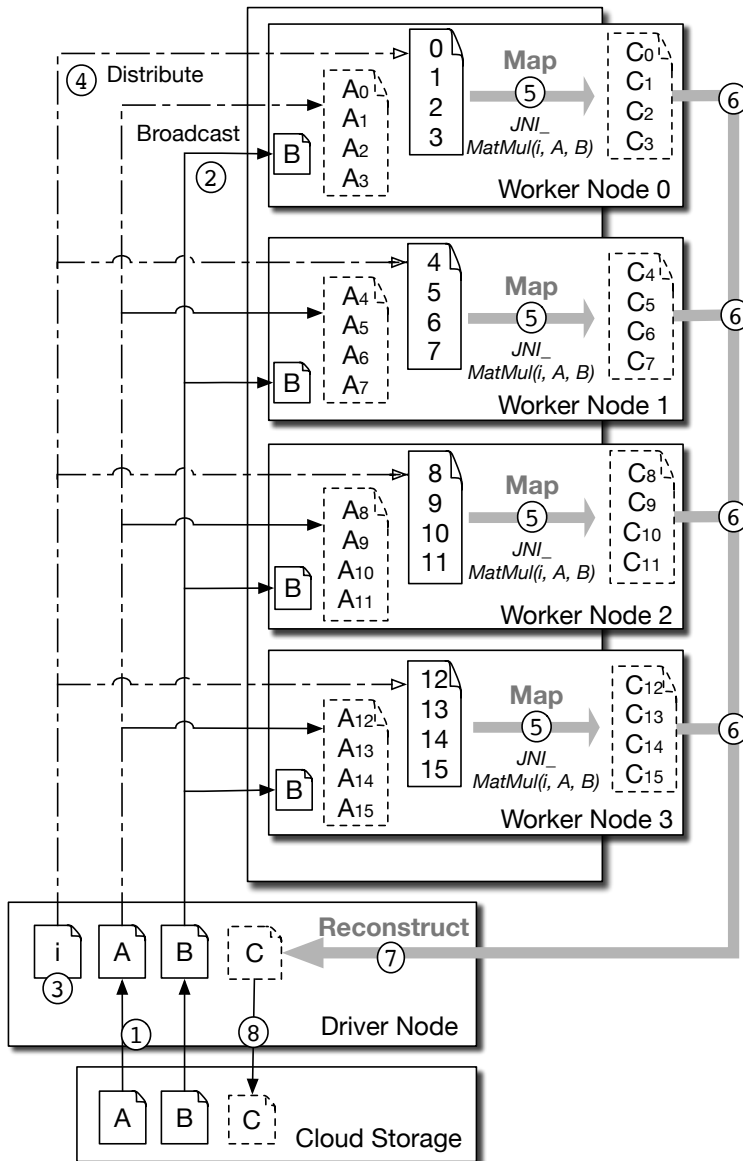
Modelo de Execução do Spark

- *Driver* node – task scheduling, resource allocation, etc
- *Worker* node – perform computation by applying map-reduce operations on large dataset
- *Resilient Distributed Dataset* – collection of data partitioned by the *driver* among the *workers* which apply parallel operations to them

Modelo de Execução do OpenMP

- *parallel for* → no loop-carried dependency between iterations
- *map (to/from)* → describe input and output variables
- *data map* → partition data according to the iteration

Corresponde ao Modelo de Execução do Spark



1. Ler as entradas (A e B) do armazenamento
2. Transmitir B
3. Gerar o conjunto dos valores tomados pelo índice de laço (i)
4. Distribuir A e i
5. Mapear a função de corpo de laço para os valores do índice do laço
6. Enviar partes de C
7. Reconstruir versão final de C
8. Escreva C no armazenamento

Código Gerado do Spark

```
// Initialization
val fs = CloudFileSystem.create(args)
val sc = new SparkContext
val info = new CloudInfo(fs, sc)

// Read each input from cloud storage
val N = 16
val A = fs.read(0)
val B = sc.broadcast(fs.read(1))

// omp parallel for
// 1 - Generate RDD of loop indexes
val RddI = sc.parallelize(0 to N-1)
val Rdd = RddI.map{ i => (i, A.slice(i*N*4, (i+1)*N*4))}
// 2 - Perform Map operations
val RddC = Rdd.map{ x =>
    (x._1, JNI_MatMul(x._1 , x._2, B.value)) }
// 3 - Merge back the results
var C = new Array[Byte](N*N*4)
RddC.collect().foreach{ x => x._2.copyToArray(C, x._1*N*4) }

// Write results back to cloud storage
fs.write(2, C)
```

Experimentação

- Caso de uso realístico
 - Host → Meu laptop conectado na Unicamp
 - Target → AWS datacenter nos EUA (North Virginia)
- Spark Cluster com 1 nó *driver* e 16 nós *worker*
 - EC2 instancias do tipo *c3.8xlarge*
 - 32 vCPU (Intel Xeon E5-2680 v2) → 16 núcleos sem HT
 - 60GB do RAM
 - Ubuntu 14.04 com Spark 2.1.0

Multiplicação de Matrizes

Matrizes 16000x16000

1GB / floating-point

Tempo de execução

Serial = 3.5h

256 núcleos = 3-8min

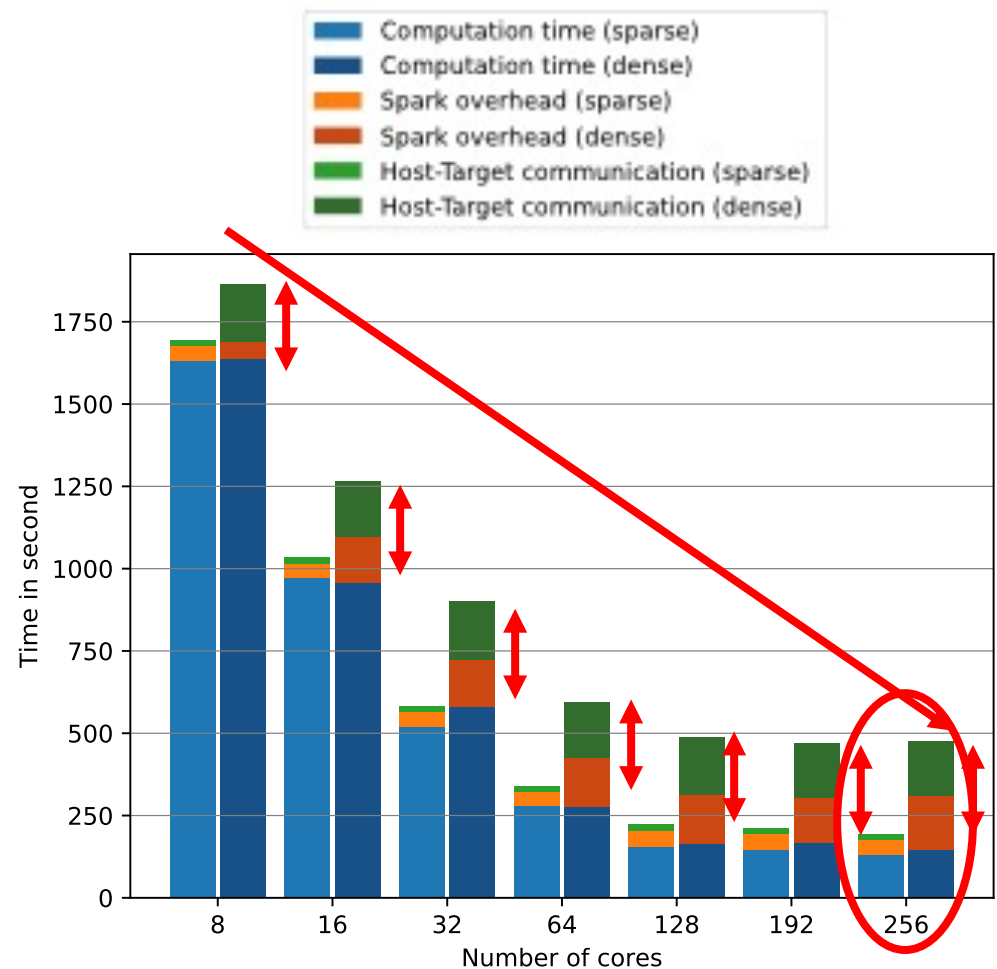
Aceleração crescendo

27x/68x em 256 núcleos

Overhead comunicação

Interno e externo

Tipo de dado importante



Limitação do Modelo de Programação

Região descarregada no nuvem

suporta

- *reduction*
- Múltiplo *parallel for*

suportará

- Blocos de código sequencial
- Um *parallel for* dentro um laço não paralelo

não suportará

- *atomic, flush, barrier, critical, or master*

Conclusão

Nova ferramenta de desenvolvimento

- Descarregamento de computação no nuvem
- Programar clusters no nuvem
- Integrar o nuvem em aplicativos locais
- Suporte qualquer fornecedor de nuvem

Modelo de programacao simples

- Suporte de C/C++ (Fortran?)
- OpenMP directivas

Experimentos preliminares

- Demonstração da viabilidade com benchmarks
- Já mostrou desempenho interessante

Trabalhos futuros

- Melhorar OmpCloud
 - “Better, Stronger, Faster”
 - Suporte dos tasks OpenMP ?
 - Segundo nível de descarregamento
 - Beta teste com os alunos
- Testar com vários domínios de aplicação
 - Aplicativos científicos
 - Renderização do Blender
 - Mobile cloud offloading ?

Obrigado! Merci!



Perguntas? Dúvidas?