Charm++

MAC5742
Computação Paralela e Distribuída

Dennis José da Silva
dennis@ime.usp.br

INSTITUTO DE MATEMATICA E ESTATÍSTICA
UNIVERSIDADE DE SÃO PAULO

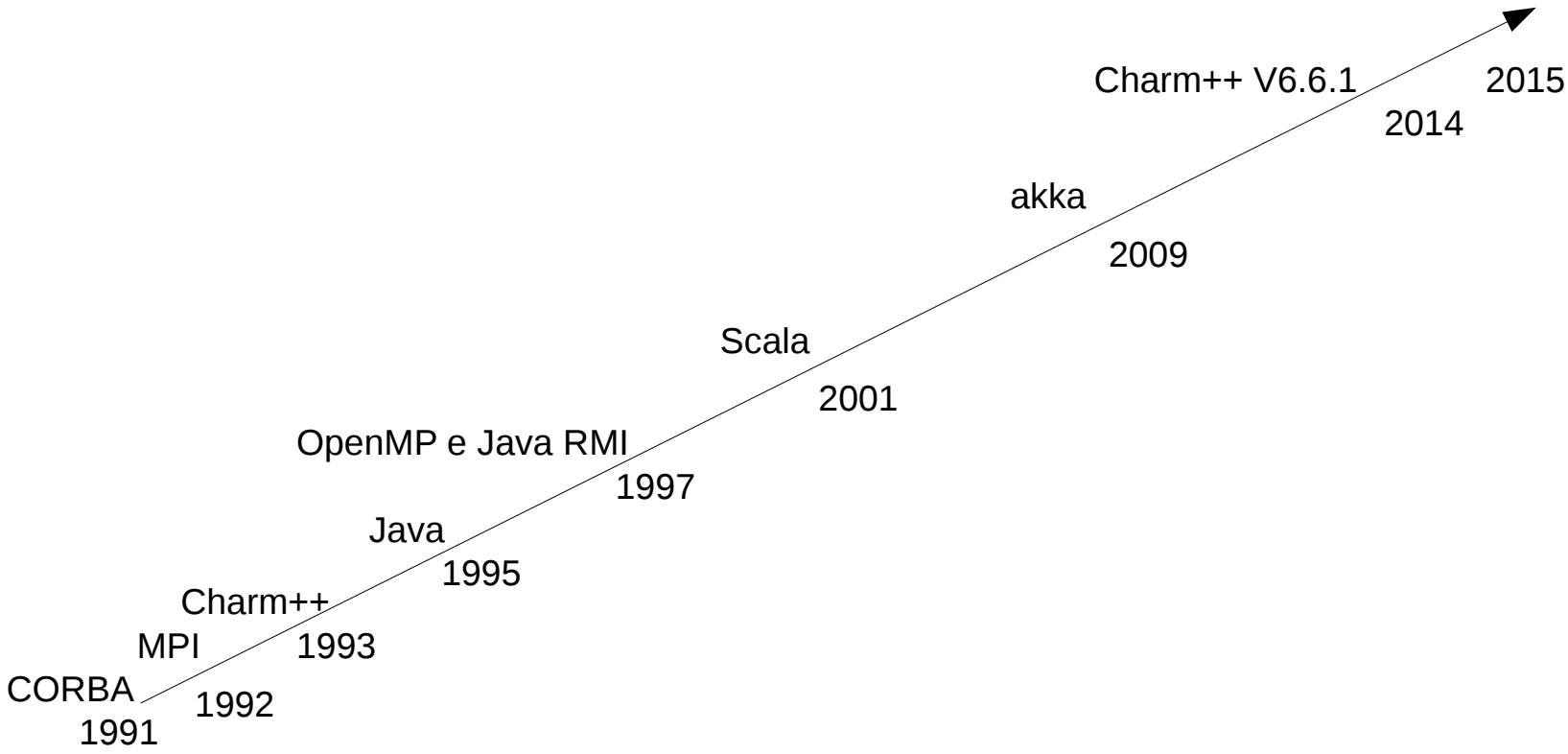
O que é Charm++?

- (Kale & Krishman, 1993) Linguagem orientada a objetos portável e paralela baseada em C++
- (Acun et. al, 2014) Sistema de programação paralela
- Desenvolvido pelo PPL (P*arallel Programming Laboratory)* da Universidade de Illinois.
- Foco no modelo MIMD (*Multiple-data Multiple-Instruction*)

Linha do Tempo

Ultimo commit do Charm++:

29/05/2015



Objetivos

- Portabilidade Eficiente (Máquina)
- Tolerância a Latência
- Balanceamento de Carga Dinâmico
- Reuso e Modularidade

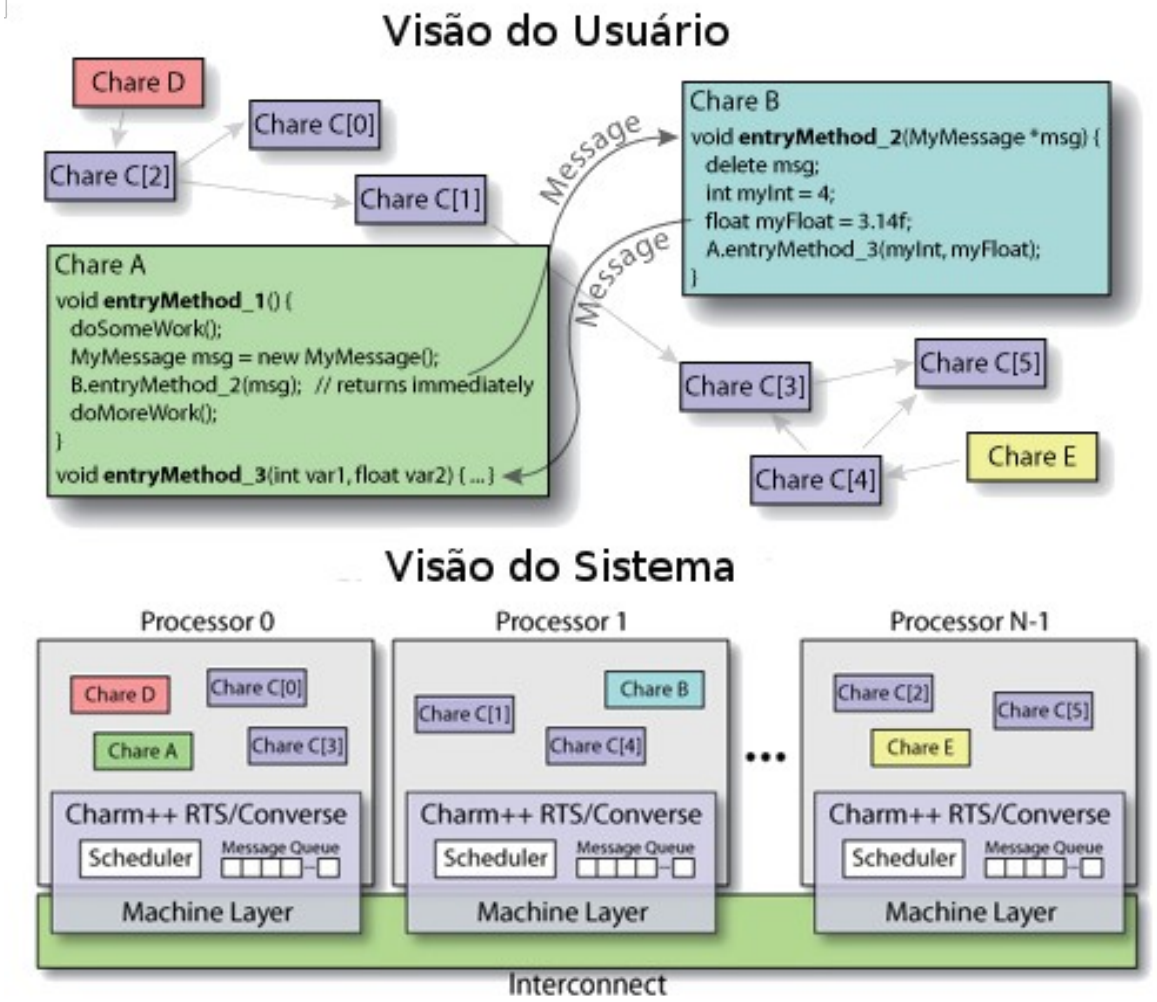
Atributos

- *Over Decomposition*
  - *Chares*
- *Asynchronous Message-Driven Execution*
  - Mensagens não bloqueantes
- *Migratability*
  - *Pup Framerwork* e RTS

RTS – *RunTime System*

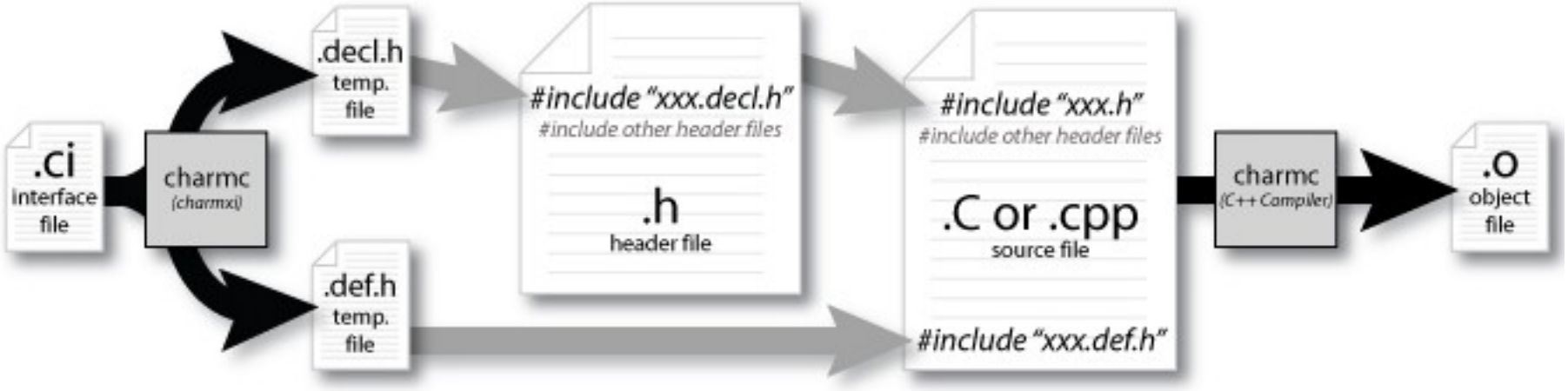
- Mapeamento de objetos *Chares* em processadores físicos
- Balanceamento de carga
- Roteirização de mensagens
- Ponto de restauração (*checkpointing)*
- Tolerância a Falhas
- Realocação dinâmica de recursos físicos

RTS – *RunTime System*



(CHARMPLUSPLUSa, 2015)

Compilação



(CHARMPLUSPLUSb, 2015)

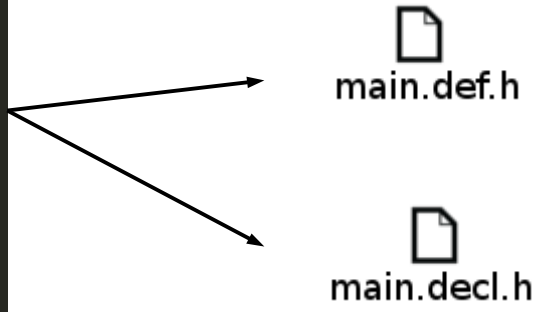
Programação

```
module MatrixCellMultiplicationChare
{
    array [1D] MatrixCellMultiplicationChare
    {
        entry MatrixCellMultiplicationChare();
        entry void multiplicate(long a[n], long b[n], int n);
    };
};
```


```
mainmodule main
{
    readonly CProxy_Main mainProxy;

    extern module MatrixCellMultiplicationChare;

    mainchare Main
    {
        entry Main(CkArgMsg *msg);
        entry void setMatrixCResult(int index, long value);
        entry void done();
    };
};
```



Programação

```
#ifndef __MATRIX_CELL_MULTIPLICATION_CHARE_H__
#define __MATRIX_CELL_MULTIPLICATION_CHARE_H__

class MatrixCellMultiplicationChare : public CBase_MatrixCellMultiplicationChare
{
public:
    MatrixCellMultiplicationChare();
    MatrixCellMultiplicationChare(CkMigrateMessage *msg);

    void multiplicate(long* a, long* b, int matrix_size);
};

#endif
```

Programação

```
#include "MatrixCellMultiplicationChare.decl.h"
#include "MatrixCellMultiplicationChare.h"
#include "main.decl.h"

extern CProxy_Main mainProxy;

MatrixCellMultiplicationChare::MatrixCellMultiplicationChare()
{}

MatrixCellMultiplicationChare::MatrixCellMultiplicationChare(CkMigrateMessage *msg)
{}

void MatrixCellMultiplicationChare::multiplicate(long* a, long* b, int matrix_size)
{
    int i = thisIndex / matrix_size;
    int j = thisIndex % matrix_size;
    long result = 0;

    for (int k = 0; k < matrix_size; ++k)
        result += a[i * matrix_size + k] * b[k * matrix_size + j];

    mainProxy.setMatrixCResult(thisIndex, result);
    mainProxy.done();
}

#include "MatrixCellMultiplicationChare.def.h"
```

Programação

```
Main::Main(CkArgMsg *msg)
{
    doneCount = 0;
    matrix_size = 64;

    if (msg->argc > 1)
        matrix_size = atoi(msg->argv[1]);


    numElements = matrix_size * matrix_size;

    matrix_a = new long[numElements];
    matrix_b = new long[numElements];
    matrix_c = new long[numElements];

    generateMatrix(matrix_a, matrix_b, matrix_c, matrix_size);


    delete msg;

    mainProxy = thisProxy;

    CProxy_MatrixCellMultiplicationChare cellMultArray =
        CProxy_MatrixCellMultiplicationChare::ckNew(numElements);

    cellMultArray.multiplicate(matrix_a, matrix_b, matrix_size);
}

Main::Main(CkMigrateMessage * msg)
{}
```

Makefile

```
CHARMDIR = CHARM++_DIR_PATH
CHARMC = $(CHARMDIR)/bin/charmc $(OPTS)

default: all
all: multmatrix

multmatrix: main.o MatrixCellMultiplicationChare.o
    $(CHARMC) -language charm++ -o multmatrix main.o MatrixCellMultiplicationChare.o

main.o: main.C main.h main.decl.h main.def.h MatrixCellMultiplicationChare.decl.h
    $(CHARMC) -o main.o main.C

main.decl.h main.def.h: main.ci
    $(CHARMC) main.ci

# Diretivas para compilar MatrixCellMultiplicationChare

clean:
    rm -f main.decl.h main.def.h main.o
    rm -f MatrixCellMultiplicationChare.decl.h MatrixCellMultiplicationChare.def.h
    rm -f MatrixCellMultiplicationChare.o
    rm -f multmatrix charmrun
```

Maleabilidade

Aplicação LeanMD rodando em um computador Stampede de 256 cores

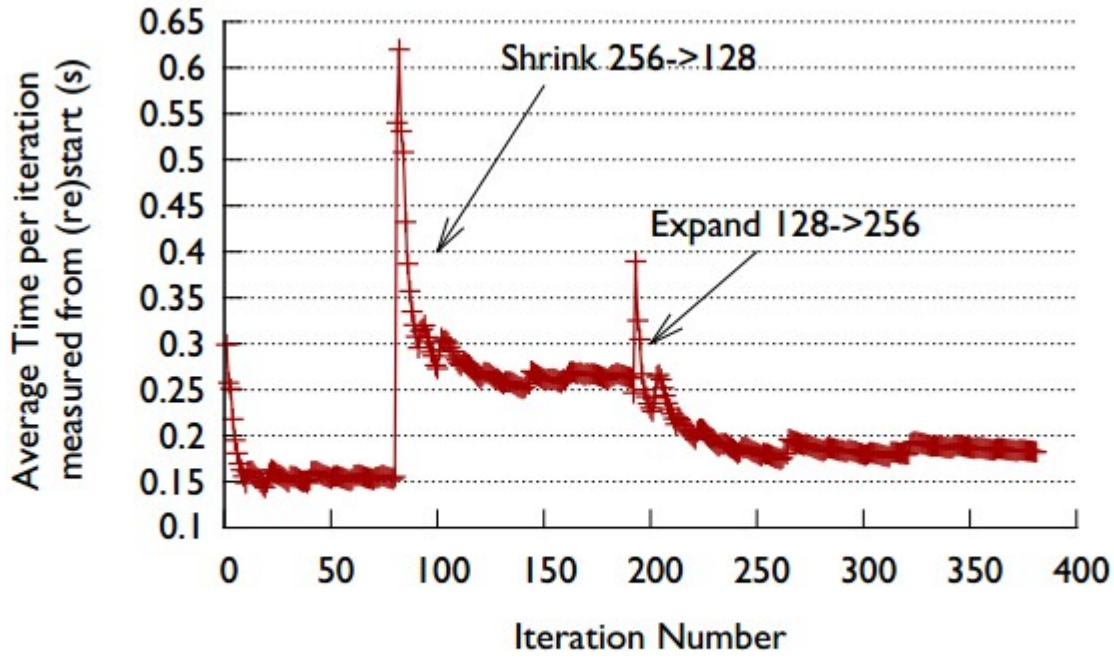


Fig. 5: Adapting load distribution on shrink and expand

(ACUN, 2014)

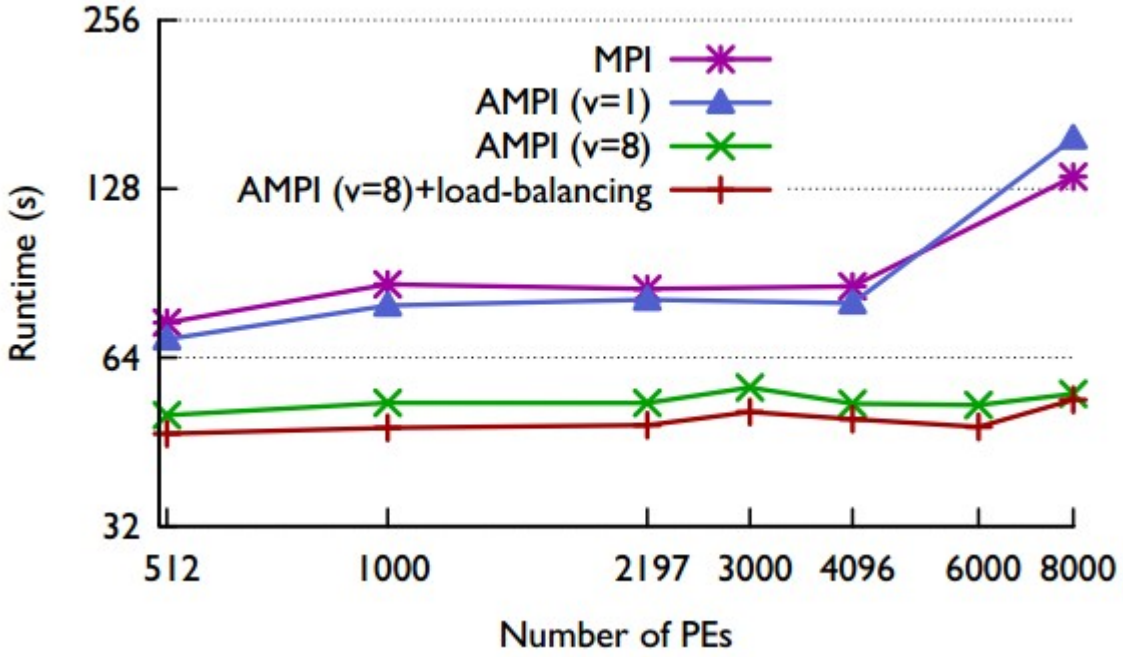
LeanMD

- Aplicativo que simula o comportamento de átomos baseado na potencia de Lennard-Jones.
- Escrita em cima do aplicativo NAMD ganhador do Gordon Bell award 2002 também escrito em Charm++
- Características
  - *Over-Decomposition:* 6D *Chare Array*
  - Balanceamento de Carga Adaptivo: Uso do RTS
  - *In-memory Checkpoint/Restart* (com simulação de falha)

LULESH

- Aplicativo que resolve equações de hidrodinâmica.

.



(ACUN, 2014)

Referências

- CHARMPLUSPLUSa, website da universidade de Illinois sobre Charm++. Disponivel em < **http://charm.cs.illinois.edu/tutorial/CharmRuntimeSystem.htm**> Acesso em 21 de Julho de 2015

- CHARMPLUSPLUSa, website da universidade de Illinois sobre Charm++. Disponivel em <**http://charm.cs.illinois.edu/tutorial/CharmComponents.htm**> Acesso em 21 de Julho de 2015

- Acun B. et al, **Parallel Programming with Migratable Objects: Charm++ in Pratice**, Proceedings of the International Conference for High Perfomace Computing, Networking, Storage and Analysis, 16-21 de Novembro de 2014, New Orleans, Louisana

- Kale, L. V., Krishnan S., Charm++: **A portable concurrent object oriented system based on C++**, University of Illinois at Urbana Champaign, Champaign, IL, 1993

Obrigado