Estudo Qualitativo e Quantitativo de Linguagens Paralelas para Arquiteturas Multicore

Mst. Raffael B. Schemmer

André L. Tibola

Junior F. Barros

Prof. Mst. Julio C. S. Anjos

Prof. Dr. Claudio F. R. Geyer

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

.inf
INSTITUTO DE INFORMÁTICA
UFRGS

Agenda

- Motivação e Objetivos.
- Linguagens e plataforma utilizada.
- Problema alvo.
- Avaliação Qualitativa.
- Avaliação Quantitativa.
- Conclusões.
- Trabalhos Futuros.

Motivação e Objetivos

- Motivação (Estudo de linguagens quanto):
  - Exploração do paralelismo.
  - Facilidade de instalação e programação.

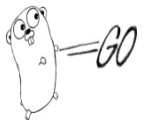
- Objetivos:
  - Estudar 4 linguagens paralelas.
  - Realizar avaliações:
  - Qualitativa (Funcionalidades).
  - Quantitativa (Desempenho e uso de recursos).

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

.inf UFRGS

Linguagens e plataforma utilizada

- Linguagens utilizadas (4):



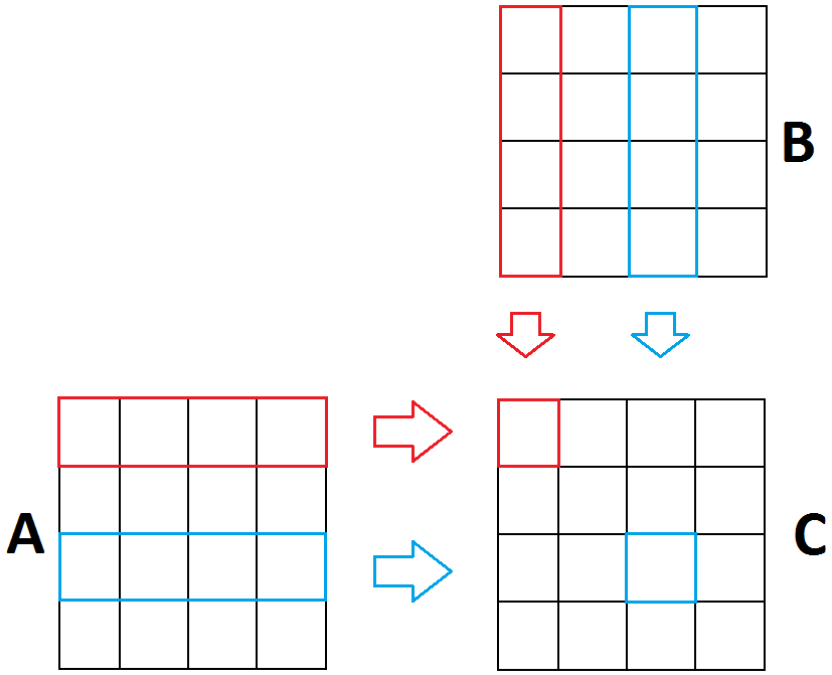
- Linguagem C com uso de POSIX Threads.
- Linguagem Go (Google).
- Linguagem Cilk Plus (Intel).
- Linguagem UPC (Gnu).
- https://github.com/RaffaelSchemmer/multicore

- Plataforma alvo:

- Processador Multicore AMD Phenom II Quad Core (2009).
- 8GBytes de memória RAM DDR3 PC1333 CL7
- Ubuntu 10.04LTS x86-64 e 12.04LTS x86-64.

Problema alvo

- Multiplicação matricial paralela de duas matrizes:
  - Ordem quadrática (2).
  - Cada thread irá calcular N linhas (Da matriz C).
  - Abordagem clássica.
  - Dados independentes compartilhados em memória.



Exemplo da multiplicação de 2 pontos de C = A x B

Avaliação Qualitativa

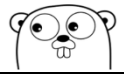
- Linguagem C:
  - Procedural/Compilada.
  - Compilador GNU GCC (Nativo Linux).
  - Certificação de padrão ANSI/ISO (C11 – 2011).
  - Possui literatura abundante (Livros e tutoriais).
  - Depende de bibliotecas para:
    - Sincronização e Concorrência (POSIX Threads).
  - Instalação e configuração do ambiente:
    - GCC 4.2 + Ubuntu 10.04LTS.
    - Compilador deve receber parâmetro -lpthread.

```
thWork = (pthread_t*)  malloc (sizeof(pthread_t) * numCores);
for(i=0,cont=0; i < numCores; i++)
{
    int **index = i;
    cont = pthread_create(&thWork[i], NULL, matrix,(void*)index);
}
i--;
pthread_join(thWork[i],(void **)&cont);
```

Criação e sincronização das Threads em C/Threads

Avaliação Qualitativa

- Linguagem Go:
  - Linguagem compilada (Compilador Go)
  - Baseada em funções e pacotes.
  - Desenvolvimento com ênfase em SVN/GIT.
  - Suporte nativo a sincronização e concorrência.
  - Instalação e Configuração:
    - SDK Go 1.2 AMD64 + Ubuntu 10.04LTS.
    - Configuração de variáveis de ambiente (Path).
    - GOMAXPROCS deve receber o número de threads.



```
i=0;
for(i < numCores){
      go matrix(i);
      i++
}

mutex.Lock();
for(finish == 0){
      mutex.Unlock();
      mutex.Lock();
}
```

Criação e sincronização das Threads em Go

Avaliação Qualitativa

- Linguagem Cilk:
  - Proposta originalmente em 1994 pelo MIT.
  - Linguagem comercial (Adquirida em 2000 pela Intel).
  - Procedural/Compilada.
  - Possui compilador (SDK) OpenSource (Cilk Plus)
  - Suporte nativo a concorrência e a sincronização.
  - Instalação e Configuração:
    - SDK Cilk Plus AMD64 + Ubuntu 10.04LTS.
    - Configuração de variáveis de ambiente (Path).

```
cilk_for (int il = 0; il < TAM; ++il)
{
    for (int jl = 0; jl < TAM; ++jl)
    {
        for (int k = 0; k < TAM; ++k)
        {
            C[il][jl] += A[il][k] * B[k][jl];
        }
    }
}
```

Exploração do paralelismo via estruturas de repetição em Cilk

Avaliação Qualitativa

- Linguagem UPC:
  - Consórcio de empresas e universidades (Berkeley):
    - Conjunto de 3 linguagens baseadas em C.
    - Este trabalho utiliza o compilador GNU UPC.
  - Suporte nativo a concorrência e o paralelismo:
    - Endereçado em nível de processo.
  - UPC implementa o conceito de memória global em máquinas distribuídas.
  - Instalação e Configuração:
    - Compilador GNU GUPC 4.9 + Ubuntu 12.04LTS.
    - Configuração de variáveis de ambiente (Path).
    - Compilador deverá receber *-fupc-threads-N*.

```
for (lin = MYTHREAD * numElements; lin < (MYTHREAD * numElements)+numElements; lin++)
{
    for (col = 0; col < TAM; col++)
    {
        for (cont = 0; cont < TAM; cont++)
        {
            C[lin][col] = C[lin][col] +  (A[lin][cont] *  B[cont][col]);
        }
    }
}
upc_barrier;
```

Programação e sincronização dos processos em UPC

Avaliação Qualitativa (Crítica)

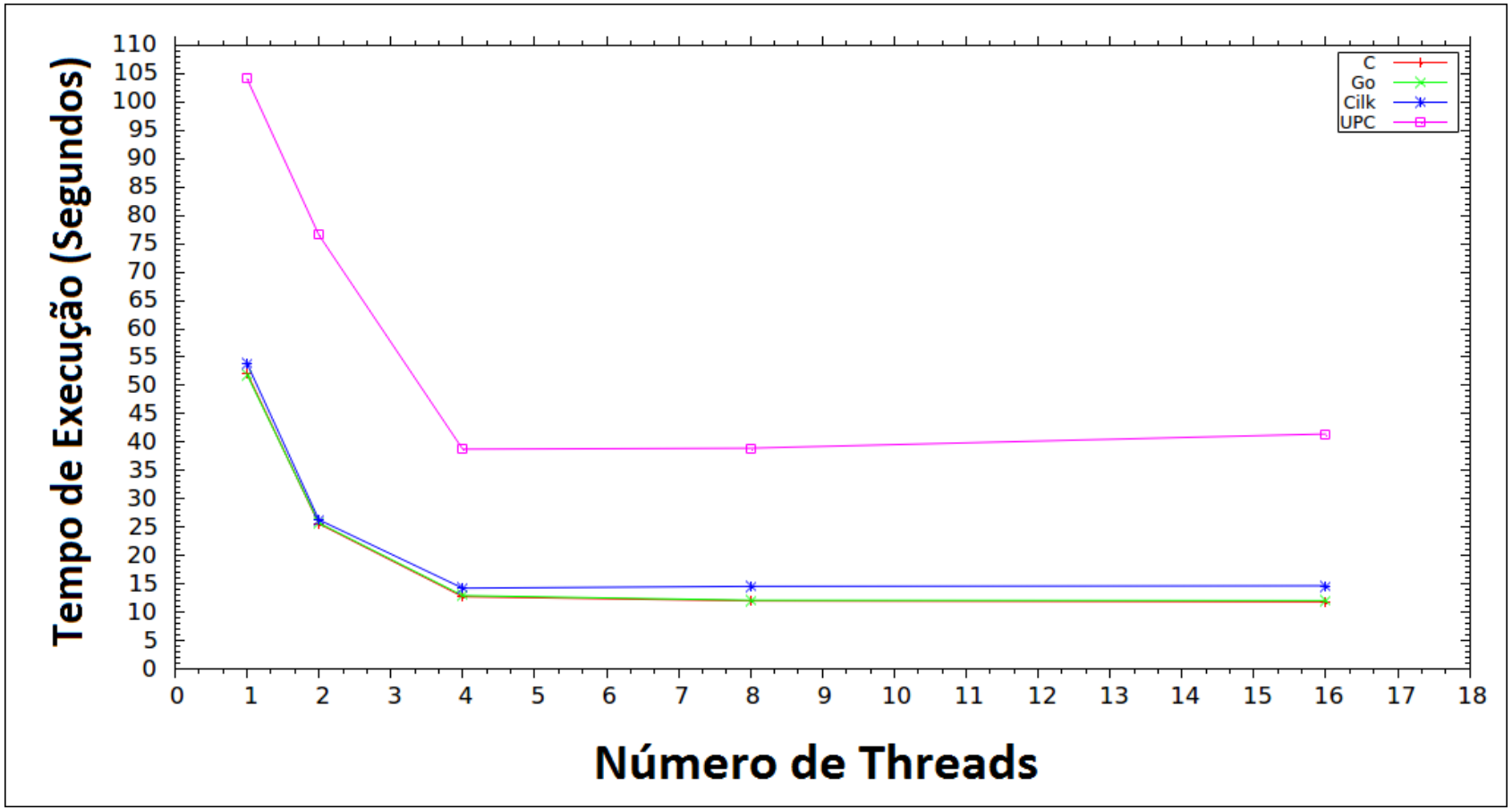
	Documentação (Configuração do Ambiente).	Acesso as ferramentas.	Configuração do ambiente.	Suporte de arquivos exemplos.	Mensagens de compilação.	Documentação (Linguagem de Programação).	Facilidade de escrita (Exploração do Paralelismo).
C	3	3	3	3	3	3	2
Go	3	3	3	3	2	3	3
UPC	2	2	2	2	3	2	2
Cilk	1	1	1	1	2	2	3

Legenda: (1) – Ruim (2) – Neutro (3) – Aceitável

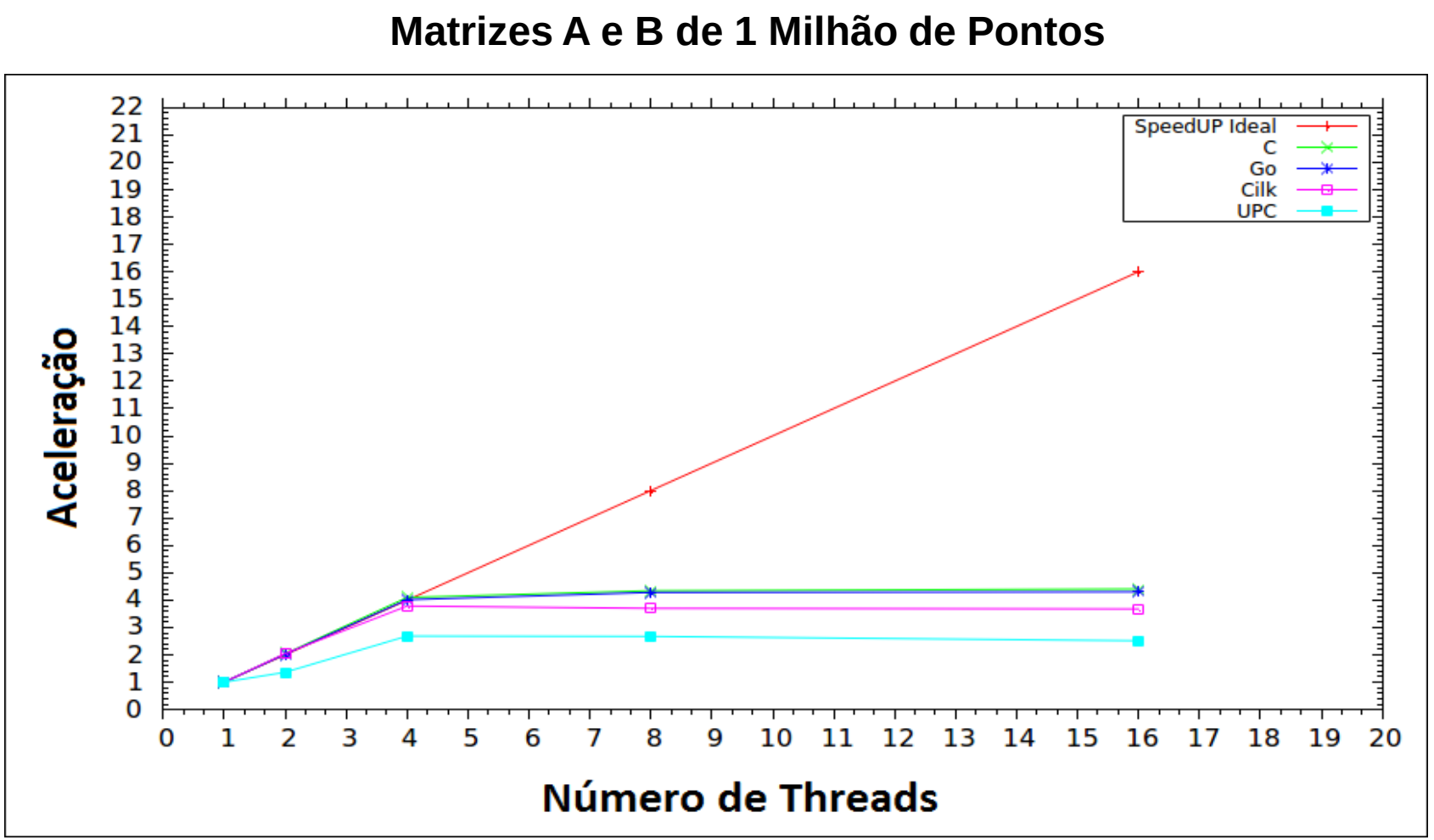
UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

.inf UFRGS

Avaliação Quantitativa (Tempo de execução)



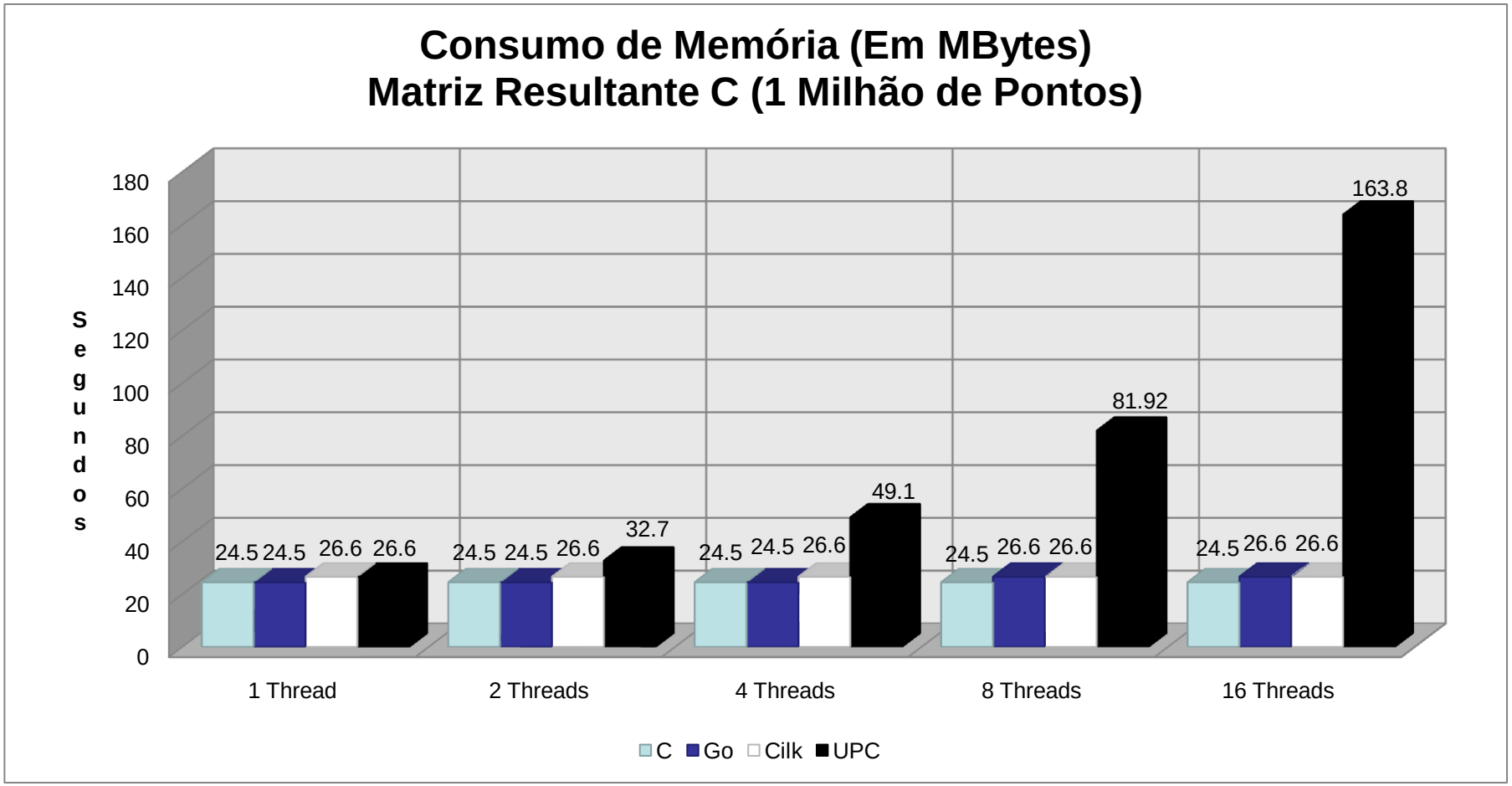
Avaliação Quantitativa (Fator de Aceleração)



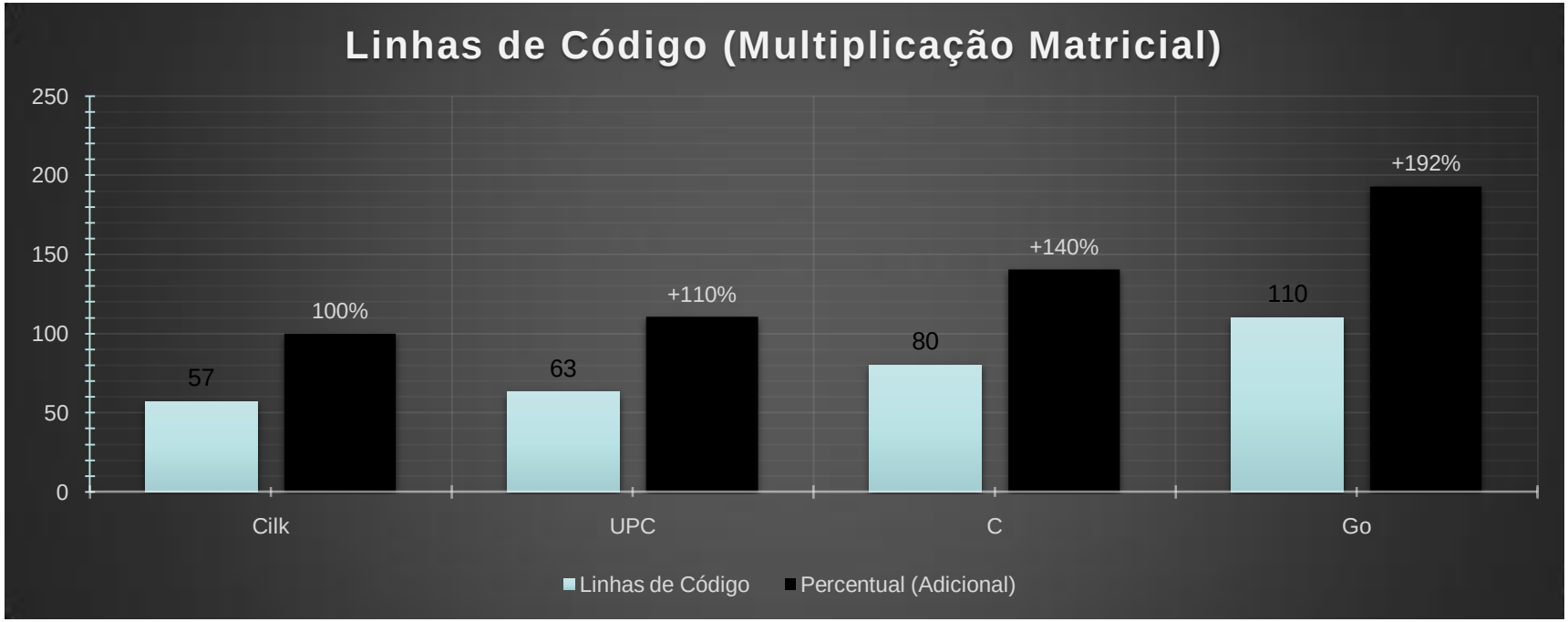
UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

.inf UFRGS

Avaliação Quantitativa (Consumo de Memória)



Avaliação Quantitativa (Linhas de código fonte)



Conclusões

- Linguagem C:
  - Obteve o melhor desempenho.
  - Exigiu a maior demanda técnica para codificação do problema.
  - Linguagem com maior facilidade de acesso e uso.
- Linguagem Go:
  - Segundo melhor desempenho (No Somatório houve um empate com C).
  - Mais bem preparada para o futuro em termos de:
    - Suporte e disponibilidade técnica.
    - Documentação e bibliotecas.
    - Facilidade de escrita.
- Linguagem Cilk:
  - Permite expressar o paralelismo de maneira intuitiva (Ex. OpenMP).
  - Linguagem com propósitos comerciais (Complexidade de acesso).
  - Utilizando threads não obteve os mesmos desempenhos que C e Go.
- Linguagem UPC:
  - Não conseguiu atingir desempenho comparado as demais linguagens avaliadas.
  - Possui limitações quanto ao consumo de memória.
  - Pouca documentação e versões específicas de compiladores.

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

.inf UFRGS

Trabalhos Futuros

- Estudar e avaliar a complexidade referente a sincronização e não apenas a concorrência.
- Uso de novos programas paralelos com demanda no uso da técnica de sincronização em sua implementação paralela.
- Avaliar os resultados utilizando flags de otimização:
  - Compilador Intel cilk explora SIMD nativamente.
- Fazer uso de novos tipos de linguagens como Java e uso da classe Thread do pacote java.lang.
- Avaliação de outros tipos de plataformas de processamento baseadas em aceleradores como Intel Xeon Phi.

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

.inf UFRGS

Referências

- [AMD 15] AMD. “AMD Phenom™ II Processors”. Disponível em: <http://www.amd.com/en-us/products/processors/desktop/phenom-ii>. Acessado em: Março 2015.

- [GCC 10] GCC. “GNU Unified Parallel C (GUPC)”. Disponível em: <https://gcc.gnu.org/projects/gupc.html>. Acessado em: Março 2015.

- [GOO 15] GOOGLE. “The Go Programming Language”. Disponível em: <https://golang.org/doc/install>. Acessado em: Março 2015.

- [INT 15a] INTEL. “The Intel® Xeon Phi™ Product Family”. Disponível em: <http://www.intel.com/content/dam/www/public/us/en/documents/product-briefs/high-performance-xeon-phi-coprocessor-brief.pdf>. Acessado em: Março 2015.

- [INT 15b] INTEL. “Intel® Xeon® Processor E5-2600 v31 Product Family” Disponível em: <http://www.intel.com.br/content/dam/www/public/us/en/documents/product-briefs/xeon-e5-brief.pdf>. Acessado em: Março 2015.

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

.inf UFRGS

Referências

- [INT 15c] INTEL. “Intel Cilk Plus Task Parallelism Tools”. Disponível em: <https://www.cilkplus.org/tutorial-cilk-tools>. Acessado em: Março 2015.
- [LLN 14] LLNL. “POSIX Threads Programming”. Disponível em: <https://computing.llnl.gov/tutorials/pthreads/>. Acessado em: Março 2015.
- [ORA 15] ORACLE. “SPARC M6-32 SERVER”. Disponível em: <http://www.oracle.com/us/products/servers-storage/servers/sparc/oracle-sparc/m6-32/sparc-m6-32-ds-2015584.pdf>. Acessado em: Março 2015.
- [PIN 14] Pink, R. “Moore's Not Enough: The Future of Computing”. Disponível em: <http://cr4.globalspec.com/blogentry/25398/Moore-s-Not-Enough-The-Future-of-Computing-by-Roger-Pink>. Acesso em: Março 2015.
- [WIK 15] WIKIPEDIA. “Matrix Multiplication”. Disponível em: <http://en.wikipedia.org/wiki/Matrix_multiplication>. Acessado em: Março 2015.

UFRGS
UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL

.inf UFRGS

Obrigado pela atenção

Perguntas?

INF/UFRGS - GPPD
raffael.schemmer@inf.ufrgs.br

UFRGS
UNIVERSIDADE FEDERAL
DO RIO GRANDE DO SUL

.inf
INSTITUTO
DE INFORMÁTICA
UFRGS