Relatório Trabalho 1

Sistemas Operacionais II
Émerson Três de Paula - 213996
Thiago Bell - 228527

Ambiente de Teste

Ubuntu Linux 16.04.1 LTS
Processador Intel i7-2600 4 x 3.40GHz (Suporte a Hyper Threading)
Memória 8 GiB DDR3
Compilador GCC 5.4.0

Casos de Teste

Matrizes de dimensões

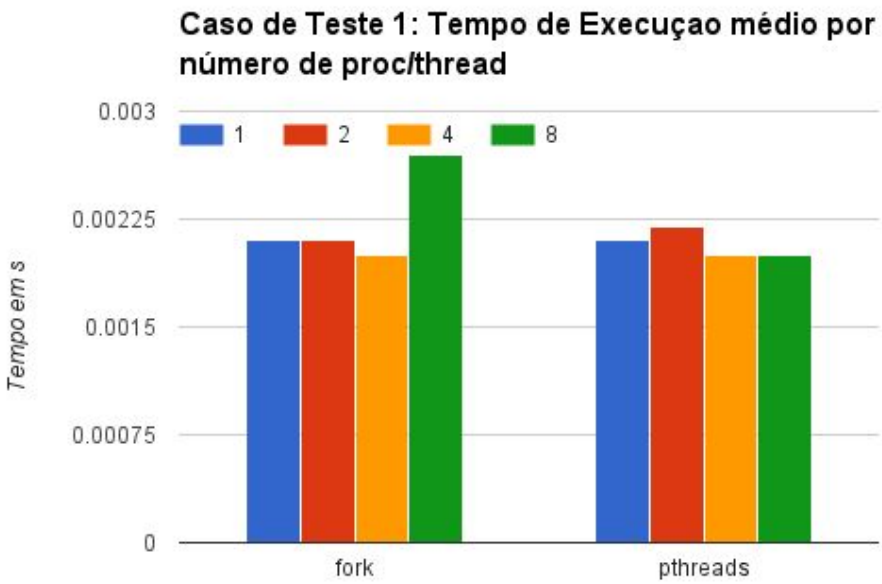
1. 10x10 e 10x10
2. 100x100 e 100x100
3. 350x700 e 700x500
4. 700x700 e 700x700
5. 1000x1000 de 1000x1000

Metodologia de Teste

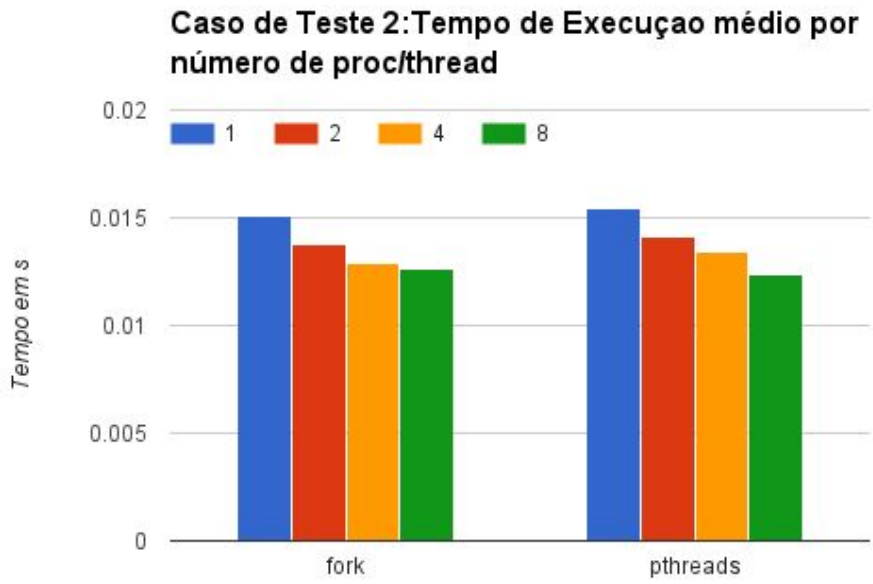
De acordo com a especificação, foram testados todos os casos de teste com as duas implementações. Para cada tripla (caso de teste, implementação, número de processos/threads), foram feitas 10 execuções e uma média aritmética foi obtida. Usou-se os seguintes números de processos/threads: 1,2,4,8. O primeiro caso foi feito para se obter um *baseline* para comparação.

Resultados

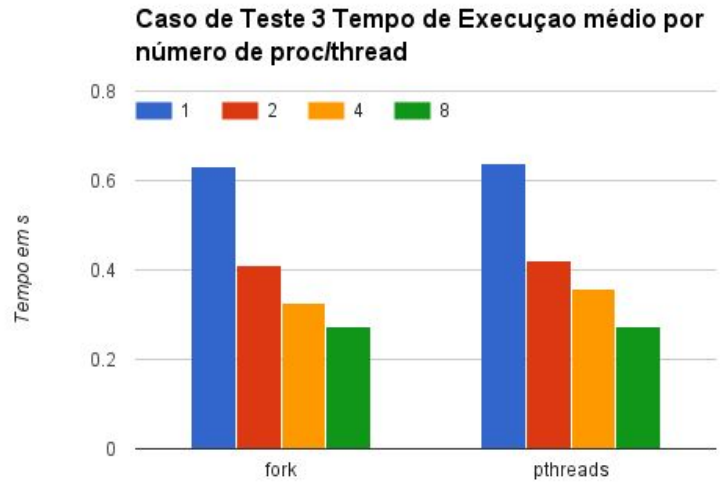
Pretendia-se com o primeiro caso de teste, mostrar que em alguns casos onde o tamanho do problema é pequeno não é vantajoso usar paralelismo, pois, os custos computacionais e temporais de gerenciar um programa paralelo são maiores do que o de executar um programa serial. No entanto, esse caso de teste não foi muito representativo. O tamanho das matrizes são tão pequenos que o tempo de execução é muito pequeno para que se obtenha uma medida de tempo adequada. Abaixo está o gráfico resultante. Cada cor de barra representa um número diferente de processos/threads. Note que o eixo vertical tem um intervalo bastante pequeno, então, as diferenças são bem pequenas.

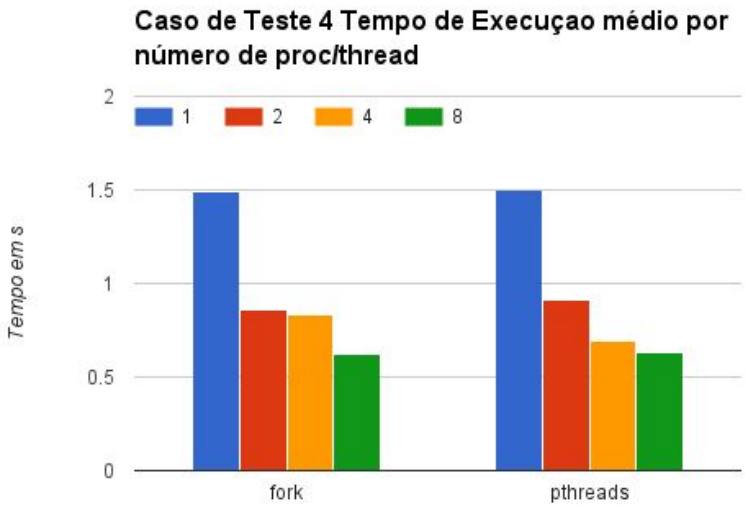


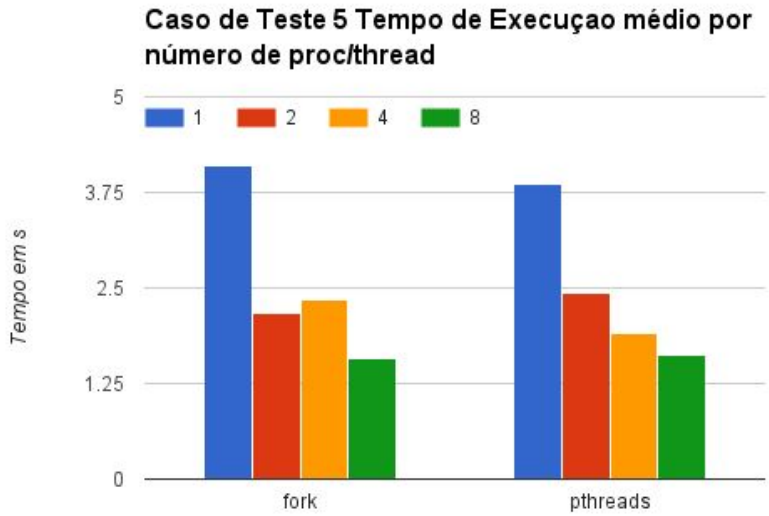
No próximo caso, obtemos uma distribuição de tempo de execução mais interessante. Um padrão de cascata (pequeno) no sentido do nível de paralelismo empregado. Nesse caso, o custo de paralelismo já vale a pena, pois há um ganho de performance. Percebe-se aqui um padrão visto nos casos de teste seguintes: A implementação fork teve um desempenho um pouco melhor que a pthreads. Comentamos o que acreditamos ser possíveis causas no final desta seção.



Para os casos seguintes, o tamanho das entradas são significativamente maiores. Neles, o efeito do paralelismo se tornou mais acentuado.







Existe um claro ganho de desempenho em relação a execução serial tanto para a versão fork quanto pthreads do programa. Isso era esperado pois o problema de multiplicação de matrizes é facilmente paralelizável e foi possível atingir um speedup razoável.
O resultado surpreendente foi que a implementação com fork foi mais eficiente que a com pthreads. Isso pode ser resultado de implementações levemente diferentes. Para o caso do fork usou-se uma área de dados compartilhada. Isso pode ter reduzido a vantagem que pthreads tinha. Alguma diferença na implementação do algoritmo de multiplicação também pode ter afetado a comparação.
Além disso, o Linux, ao realizar um fork de um processo não duplica todas as suas páginas imediatamente, mas permite que os processos dividem as mesmas desde que não haja modificação dos dados. Isso também pode ter colaborado para esse resultado.

Comunicação Interprocesso na Versão Fork

No caso do programa com fork, criou-se uma área de memória compartilhada onde os processos escrevem os resultados das operações em cada linha. Nessa mesma área, é armazenada as tarefas a serem realizadas por cada subprocesso. Nesse caso, o intervalo de linhas a processar. Ao término dos processos filhos, o processo pai copia o resultado dessa área de dados para uma área própria e escreve os resultados em arquivo.