| | | | | |
|---|---|---|---|---|
| tainah.silva@ccc.ufcg.edu.br | https://github.com/tainahemmanuele/programacao_concorrente | JOAO MARCOS ARISTIDES ALMEIDA | SAULO SAMUEL FERREIRA DO NASCIMENTO | TAINAH EMMANUELE SILVA |

**NOTA 5.9 (conversar com os alunos sobre as questões em clang)**

**Q1 - 0.5/1.0**
- código ok
- resultados estranho e explicação ruim. rodou só uma vez?

**Q2 - 2,0/2.0**

**Q3 - 1.5/2.0**

a1) java 0.25/0.5
- Funciona
- gateway não precisava ser uma thread
    - linhas 28/30 muito esquisitas
    - pq o sync no acesso ao canal na linha 28?
- em Request
    - linhas 26/28 esquisitas
    - na linha 31 qual a razão da trava no acesso ao channel? estranho
- qual a razão de existir a classe lock? o canal deveria resolver tudo. usa canais de maneira errada dentro de lcol

a2) clang 0.5/0.5
- ok

b1) java 0.25/0.5
- Funciona
- implementação de canal muito estranha. o receive só entrega se o canal estiver cheio? só faz send até encher? depois de receber, não consome o item?
- mesmo problemas apontados nas classes acima

b2) clang 0.5/0.5
- ok

**Q4 1.4/2.0**

a1) java 0.25/0.5
- Funciona
- mesmos problemas de design apontados para a Q3

a2) clang 0.4/0.5
- não termina no timeout

b1) java 0.25/0.5
- funciona
- mesmo problemas de design apontados para a q3

b2) clang 0.5/0.5
- ok

**Q5 0.5**
- parte A

    - usou a mesma instância da JVM para todos os experimentos (um exp. pode ser impactado pelo anterior)
    - plots não mostram a variabilidade
    - ploto o custo por operação, ao invés de vazão ou tempo total
    - variações estranha na métrica (linha azul) grafico_letra_a_write_0.2.png quedas e subidas
    - estranho, nos ultimos plots, o synchronizedmap ser melhor
- parte B
    - mesmos problemas da parte A

---

| | | | | |
|---|---|---|---|---|
| marcus.tenorio@ccc.ufcg.edu.br | https://github.com/dialmformart/concprog | MARCUS ANTONIO ROCHA TENORIO | KAIO KASSIANO MOURA OLIVEIRA | |

**NOTA 6.3**

**Q1 - 1.5**
- relatório decente. seria bom se tivessem complementado com várias execuções do experimento para termos ideia da variabilidade

**Q2 - 2.0**

**Q3 - 1.8**

a1) java 0.4/0.5
- funciona
- uma  mistura estranha de mutex e variáveis condicionais. pq não usar o channel da questão anterior?
- seria melhor que as retardatárias parassem de executar após o fim da primeira.

a2) clang 0.5/0.5
- funciona (precisei mudar a ordem dos métodos para poder compilar, mas … ok)

b1) java 0.4/0.5
- funciona
- mesma mistura estranha de Q3-a1. pq nao usar o canal. Outra coisa, me parece que o código seria mais simples se não tivesse sido adaptado de Q3-a1

b2) clang 0.5/0.5
- funciona (precisei mudar a ordem dos métodos para poder compilar, mas … ok)

**Q4 0.0**

a1) java 0.0/0.5
- o cancelamento deveria ser baseado de fato no timeout ao invés do número sorteado.

a2) clang 0.0/0.5
- o cancelamento deveria ser baseado de fato no timeout ao invés do número sorteado.

b1) java 0.0/0.5

- o cancelamento deveria ser baseado de fato no timeout ao invés do número sorteado.

b2) clang 0.0/0.5

- o cancelamento deveria ser baseado de fato no timeout ao invés do número sorteado.

**Q5 1.0**

- parte A
  - tem algo muito errado com os experimentos de mapas. Não há diferença entre as estruturas e o desempenho melhora com o aumento do número de threads.
- parte B
  - ok

---

| | | | | |
|---|---|---|---|---|
| mattheus.rodrigues@ccc.ufcg.edu.br | https://github.com/MattheusB/paradigma-concorrente | ANARCO QUARESMA ZEFERINO NASCIMENTO | JOSE MANOEL DOS SANTOS FERREIRA | MATTHEUS BRITO RODRIGUES |

**NOTA 6.0**

**Q1 - 1.2/2.0**

- relatório ok. seria bom se tivessem complementado com várias execuções do experimento para termos ideia da variabilidade

**Q2 - 2.0 / 2.0**

**Q3 - 1.1/2.0**

a1) java 0.0/0.5

- estranha a função gateway. um notify:32 com um wait:34 logo depois. para quê isso?
- não consegui observar concorrência. parece que só uma thread é colocada para executar.

a2) clang 0.5/0.5

- funciona

b1) java 0.3/0.5

- funciona
- consumer não precisava ser um runnable (não é executado em uma thread diferente)
- pq não usou o canal de q2?

b2) clang 0.5/0.5

- funciona

**Q4 1.0/2.0**

a1) java

- onde está o timeout?

a2) clang
b1) java
- onde está o timeout?

b2) clang
**Q5 0.5/2.0**
- parte A
  - não indica variabilidade
  - plot ruim, não tem label no eixo x
  - subida e queda estranhas no segundo plot
  - não explica as métricas
- parte B
  - não indica variabilidade
  - plot ruim, não tem label no eixo x
  - não discute tamanho do mapa nem proporção de operações
  - não explica as métricas

---

| | | | |
|---|---|---|---|
| hiago.sousa@ccc.ufcg.edu.br | https://github.com/hiagonfs/lista01-progconcorrente | HIAGO NATAN FERNANDES DE SOUSA | GABRIELA MOTTA OLIVEIRA |

**NOTA 3.95**

**Q1 - 0.2/2.0**
- explicação ruim. sem comprovação experimental

**Q2 - 2.0/2.0**
**Q3 - 0.75/2.0**
a1) java 0.5/0.5
- funciona
- muito bom. usou o canal da primeira questão

a2) clang 0.0/0.5
- Não fez

b1) java 0.25/0.5
- funciona +- (não esperou todas as threads em um teste que fiz)
- bacana. usaram o canal da primeira questão
- não deveria precisar do join na linha 29

b2) clang 0.0/0.5
- Não fez

**Q4 0.5/2.0**
a1) java 0.25/0.5
- funciona +- (o programa não para ao fim do primeiro request, embora escreva na saída padrão uma indicação que este finalizou)

- estranho a utilização de um timeout externo. há maneiras mais elegantes de fazer isso.

a2) clang 0.0/0.5
- não fez

b1) java 0.25/0.5

funciona +- (o programa não para com o timeouts de 16s)
- estranho a utilização de um timeout externo. há maneiras mais elegantes de fazer isso.

b2) clang 0.0/0.5
- não fez

**Q5 0.5/2.0**
- parte A
  - ruim. o importante seria investigar o efeito da contenção
- parte B
  - ruim. o importante seria investigar o efeito da contenção

---

| | | | | |
|---|---|---|---|---|
| amanda.costa@ccc.ufcg.edu.br | https://github.com/dalesEwerton/PC-Lista1 | AMANDA VIVIAN ALVES DE LUNA E COSTA | AMINTAS VICTOR RAMOS PEREIRA | DALES EWERTON LOPES FRAGOSO |

**NOTA 9,6**

**Q1 - 1.8/2.0**
- relatório bom. experimento com variação da contenção. faltou mostra variabilidade dos resultados.

**Q2 - 2.0/2.0**
- ok

**Q3 - 2.0/2.0**

a1) java 0.5/0.5
- muito bom. usando o canal da Q2

a2) clang 0.5/0.5
- ok

b1) java 0.5/0.5

- muito bom. muito simples

b2) clang 0.5/0.5
- ok

**Q4 /2.0**

a1) java 0.25/0.5
- hum … não tão bacana colocar uma flag explícita para timeout. pq não implementar um close?

a2) clang 0.5/0.5
- ok

b1) java 0.25/0.5
- hum … não tão bacana colocar uma flag explícita para timeout. pq não implementar um close?

b2) clang 0.0/0.5
- não termina

**Q5 1.8/2.0**
- parte A
    - eixos invertidos no plot
- parte B
    - eixos invertidos no plot

---

| | | | | |
|---|---|---|---|---|
| jose.nunes@ccc.ufcg.edu.br | https://github.com/Benardi/concurrent_prog | JOSE BENARDI DE SOUZA NUNES | GUSTAVO DINIZ MONTEIRO | RUAN ROBERTO ELOY SILVEIRA |

QUE REPOSITÓRIO BAGUNÇADO DO CAC3T3

**NOTA 8.6**

**Q1 - 2.0/2.0**
- muito bom

**Q2 - 2.0/2.0**
- **ok**

**Q3 - 1.3/2.0**

a1) java 0.0/0.5
- há uma condição de corrida importante no código. a linha 16 o Main pode perder uma observação e o valor do Requester ser trocado mais de uma vez
- implementação ruim. busywait

a2) clang 0.5/0.5
- ok

b1) java 0.3/0.5
- funciona
- implementação mais complexa do que deveria

b2) clang 0.5/0.5
- ok

**Q4 1.6/2.0**

a1) java 0.5/0.5
- funciona
- solução muito complicada. busy wait

a2) clang 0.3/0.5
- ok

b1) java 0.3/05
- funciona
- solução muito complicada. busy wait

b2) clang 0.5/0.5

- ok
- deus! pq declararam timespec?

**Q5 1.7/2.0**

- parte A
  - ok. bom trabalho. mas a apresentação poderia ser melhor. p.ex comparar map vs map e list vs list (ao invés de colocar todas as struct no mesmo plot)
- parte B
  - ok. bom trabalho. mas a apresentação poderia ser melhor. p.ex comparar map vs map e list vs list (ao invés de colocar todas as struct no mesmo plot)

---

| | | | | |
|---|---|---|---|---|
| filipe.lima@ccc.ufcg.edu.br | https://github.com/filipegl/lista-1-prog-concorrente | FILIPE GOMES DE LIMA | JOSE THIAGO DOS SANTOS SILVA | LUCAS BARROS ROCHA |

**NOTA 6,7**

**Q1 - 1.2/2.0**

- relatório ok. seria bom se tivessem complementado com várias execuções do experimento para termos ideia da variabilidade

**Q2 - 2.0/2.0**

**ok**

**Q3 - 2.0/2.0**

a1) java 0.5/0.5

- funciona
- ok

a2) clang 0.5/0.5

- funciona
- ok

b1) java 0.5/0.5

- funciona
- ok

b2) clang 0.5/0.5

- funciona
- ok

**Q4 1.0/2.0**

a1) java 0.5/0.5

- funciona
- ok

a2) clang 0.0/0.5

- está sempre retornando 0 como primeiro valor

b1) java 0.5/0.5

b2) clang 0.0/0.5

- porque uma nova thread somente para criar as demais?

- não funciona. não dorme

**Q5 0.5/2.0**

- parte A 0.2/1.0
  - faltou discutir os resultados
  - não mostra variabilidade
  - resultados contra-intuitivos. mapas mais rápidos conforme se aumenta a contenção
- parte B 0.3/1.0
  - não mostra variabilidade
  - resultados contra-intuitivos. mapas mais rápidos conforme se aumenta a contenção

---

| | | | | |
|---|---|---|---|---|
| vinicius.jorge.silva@ccc.ufcg.edu.br | https://github.com/viniciusjps/pc-lista1 | VINICIUS JORGE PEREIRA DA SILVA | HEMILLAINY SUELLEN SOUSA SANTOS | SAMMARA BESERRA NUNES |

**NOTA 2.3**

**Q1 - 1.2/2.0**

- relatório ok. seria bom se tivessem complementado com várias execuções do experimento para termos ideia da variabilidade

**Q2 - 0.5/2.0**

- ruim. as primitivas do canal deveriam prover sincronização, ao invés desta responsabilidade estar no cliente

**Q3 - 0.3/2.0**

a1) java 0.1/0.5

- funciona
- extends Thread não é uma boa ideia
- gateway deveria retornar o valor retornado por uma das execuções de request (ao invés do Id da thread)
- implementação ruim. busy wait bastante complicado

a2) clang 0.0/0.5

- não fizeram

b1) java 0.2/0.5

- funciona
- extends Thread não é uma boa ideia
- implementação ruim. busy wait bastante complicado

b2) clang 0.0/0.5

- não fizeram

**Q4 0.3/2.0**

a1) java 0.1/0.5

- funciona
- extends Thread não é uma boa ideia
- gateway deveria retornar o valor retornado por uma das execuções de request (ao invés do Id da thread)
- implementação ruim. busy wait bastante complicado

a2) clang
- não fizeram

b1) java 0.2/0.5
- funciona
- extends Thread não é uma boa ideia
- implementação ruim. busy wait bastante complicado

b2) clang
- não fizeram

**Q5 0.0/2.0**
- parte A - não fizeram
- parte B - não fizeram

---

| | | | | |
|---|---|---|---|---|
| felipe.calixto@ccc.ufcg.edu.br | https://github.com/felipeemerson/prog-conc-lista1 | FELIPE EMERSON DE OLIVEIRA CALIXTO | WESLEY LUCENA QUEIROZ | |

**NOTA 7.0**

**Q1 - 1.2/2.0**
- relatório ok. seria bom se tivessem complementado com várias execuções do experimento para termos ideia da variabilidade

**Q2 - 2.0/2.0**
- ok

**Q3 - 1.4/2.0**

a1) java 0.2/0.5
- funciona
- implementação ruim. busy wait bastante complicado

a2) clang 0.2/0.5
- funciona
- solução com busy wait. ruim

b1) java 0.5/0.5
- funciona

b2) clang 0.5/0.5
- funciona

**Q4 0.8/2.0**

a1) java 0.2/0.5
- funciona

- solucao pouco elegante (sleep)

a2) clang 0.2/0.5
- solução ruim. busy wait

b1) java 0.2/0.5
- funciona
- solucao pouco elegante (sleep)

b2) clang 0.2/0.5
- solução ruim. busy wait

**Q5 1.6/2.0**
- parte A 0.8
  - bacana. organizado
  - não mostra variabilidade nos plots.
  - seria melhor plotar struct1 vs struct2 nos gráticos
- parte B 0.8
  - bacana. organizado
  - não mostra variabilidade nos plots.
  - seria melhor plotar struct1 vs struct2 nos gráticos

---

| | | | | |
|---|---|---|---|---|
| gabriel.almeida.azevedo@ccc.ufcg.edu.br | https://github.com/cassioegc/lista1-concorrente | GABRIEL ALMEIDA AZEVEDO | CASSIO EDUARDO GABRIEL CORDEIRO | GEOVANE DO NASCIMENTO SILVA |

**NOTA 7.0**

**Q1 - 2.0/2.0**
- bom relatório

**Q2 - 0.5/2.0**
- ruim. as primitivas do canal deveriam prover sincronização, ao invés desta responsabilidade estar no cliente

**Q3 - 1.75/2.0**

a1) java 0.25/0.5
- conditional vars em java precisam ser usadas com um idioma bem definido que evita o spurious wake-up

a2) clang 0.5/0.5
- ok

b1) java 0.5/0.5
- ok

b2) clang 0.5/0.5
- ok

**Q4 1.75/2.0**

a1) java 0.25/0.5

- conditional vars em java precisam ser usadas com um idioma bem definido que evita o spurious wake-up

a2) clang 0.5/0.5

- ok

b1) java 0.5/0.5

- ok

b2) clang 0.5/0.5

- ok

**Q5 1.0/2.0**

- parte A 0.5/1.0
  - +- organizado
  - não mostra variabilidade nos plots.
  - seria melhor plotar struct1 vs struct2 nos gráficos, além de juntar em um mesmo plot a variação da concorrência
- parte B 0.5/1.0
  - +- organizado
  - não mostra variabilidade nos plots.
  - seria melhor plotar struct1 vs struct2 nos gráficos, além de juntar em um mesmo plot a variação da concorrência

---

| | | | |
|---|---|---|---|
| ronan.souza@ccc.ufcg.edu.br | https://github.com/yurikelvin/lista_fpc | RONAN DE ARAUJO SOUZA | YURI KELVIN MOURA SOUSA E SILVA |

**NOTA 8.2**

**Q1 - 1.2/2.0**

- relatório ok. seria bom se tivessem complementado com várias execuções do experimento para termos ideia da variabilidade

**Q2 - 2.0/2.0**

- **ok**

**Q3 - 2.0/2.0**

a1) java 0.5/0.5

- ok. bacana, usou o chan

a2) clang /0.5

- ok, funciona

b1) java 0.5/0.5

- ok. bacana. usou o chan

b2) clang 0.5/0.5

- ok, funciona

**Q4 2.0/2.0**

a1) java 0.5/0.5

- ok. bacana, usou o chan

a2) clang 0.5/0.5
- ok, funciona

b1) java 0.5/0.5
- ok. bacana, usou o chan

b2) clang 0.5/0.5
- ok, funciona

**Q5 1.0/2.0**
- parte A 0.5/1.0
  - não mostra variabilidade
  - resultados contra-intuitivos. mapas mais rápidos conforme se aumenta a contenção
- parte B 0.5/1.0
  - não mostra variabilidade
  - resultados contra-intuitivos. mapas mais rápidos conforme se aumenta a contenção

---

| | | | |
|---|---|---|---|
| david.quaresma@ccc.ufcg.edu.br | https://github.com/dfquaresma/fpc-lista1 | DAVID FERREIRA QUARESMA | RENATO DANTAS HENRIQUES |

**NOTA 9.5**

**Q1 - 2.0/2.0**
- bom relatório

**Q2 - 2.0/2.0**
- **ok**

**Q3 - 2.0/2.0**

a1) java 0.5/0.5
- bacana. usou um canal

a2) clang 0.5/0.5
- ok

b1) java 0.5/0.5
- ok

b2) clang 0.5/0.5
- ok

**Q4 1.5/2.0**

a1) java 0.5/0.5
- ok

a2) clang 0.5/0.5

b1) java 0.0/0.5
- spurious wakeup na linha 33?

b2) clang 0.5/0.5

**Q5 2.0/2.0**

- parte A /1.0
    - ok
- parte B /1.0
    - ok

---

| | | | | |
|---|---|---|---|---|
| lucas.oliveira@ccc.ufcg.edu.br | https://github.com/LucasFOliveira/pc | LUCAS FERNANDES DE OLIVEIRA | PEDRO HENRIQUE COSTA MAIA | WESLEY SANTOS SILVA |

**NOTA 4.8**

**Q1 - 0.2/2.0**
- explicação ruim. rodaram uma única vez

**Q2 - 2.0/2.0**
- ok

**Q3 - 0.5/2.0**

a1) java 0.5/0.5
- ok

a2) clang 0.0/0.5
- não fez

b1) java 0.0/0.5
- não usam timeout. Ao invés disso, dormem pelo tempo que "acham" que as threads irão gastar no seus sleeps

b2) clang 0.0/0.5
- não fez

**Q4 - 0.5/2.0**

a1) java 0.5/0.5
- ok

a2) clang 0.0/0.5
- não fez

b1) java 0.0/0.5
- não fez

b2) clang 0.0/0.5
- não fez

**Q5 - 1.6/2.0**
- parte A 0.8/1.0
    - resultados ok
    - deveriam ter plotado a variabilidade dos resultados.
- parte B 0.8/1.0
    - resultados estranhos
    - deveriam ter plotado a variabilidade dos resultados.

| | | | |
|---|---|---|---|
| paulo.felipe.silva@ccc.ufcg.edu.br | https://github.com/paulofelipefeitosa/concurrent-programming | FELIPE MOTA DOS SANTOS | PAULO FELIPE FEITOSA DA SILVA |

**NOTA 8,95**

**Q1 - 2.0/2.0**
- muito bom

**Q2 - 2.0/2.0**
- ok

**Q3 - 1.75/2.0**

a1) java 0.25/0.5
- spurious wakeup

a2) clang 0.5/0.5
- muito bom

b1) java 0.5/0.5
- ok

b2) clang 0.5/0.5
- ok

**Q4 2.0/2.0**

a1) java
- ok

a2) clang 0.5/0.5
- ok

b1) java
- ok

b2) clang 0.5/0.5
- ok

**Q5 1.2/2.0**
- parte A 0.6
  - deveriam ter plotado a variabilidade dos resultados.
  - métrica não muito boa
- parte B 0.6
  - deveriam ter plotado a variabilidade dos resultados.
  - métrica não muito boa

| | | | | | |
|---|---|---|---|---|---|
| hugo.galvao@cc.ufcg.edu.br | https://github.com/KlebersonCanuto/ListasProgConc | HUGO ADDOBBATI GALVAO | KLEBERSON MATHEUS CUNHA SILVA CANUTO | JOSE DE ARIMATEIA MORAIS FILHO | |

**NOTA 3.5**

**Q1 - 0.0/2.0**
- sem experimentos, sem análise

**Q2 - 2.0/2.0**
- ok

**Q3 - 1.0/2.0**

a1) java 0.25/0.5
- spurious wakeup :61

a2) clang 0.25/0.5
- as funções request não retornam o valor correspondente ao tempo que dormiram

b1) java 0.5/0.5

b2) clang 0.0/0.5
- qual a razão de existir do finishedThreads?

**Q4 0.5/2.0**

a1) java 0.25/0.5
- spurious wakeup :87

a2) clang 0.25/0.5
- as funções request não retornam o valor correspondente ao tempo que dormiram

b1) java 0.0/0.5
- super-complicado
- sleep fixo nas threads de request

b2) clang 0.0/0.5
- não fizeram

**Q5 0.0/2.0**
- parte A
  - não fizeram
- parte B
  - não fizeram

---

**NOTA**

**Q1 - /2.0**

**Q2 - /2.0**

**Q3 - /2.0**

a1) java /0.5

a2) clang /0.5

b1) java /0.5

b2) clang /0.5

**Q4 /2.0**

a1) java

a2) clang

b1) java

b2) clang

**Q5 /2.0**

- parte A
- parte B