

Análise de Paralelização em Contagem de Acessos de CDN

Enrico Minto Spanier - 10419775

1. Introdução

Processamos 10.000.000 linhas de log sobre 100.000 URLs de um manifesto para identificar acessos (hits) em cenário típico de CDN. São avaliadas três abordagens: versão sequencial (referência) e duas paralelas com OpenMP usando sincronização por `critical` e por `atomic`.

Estratégias de Sincronização

- critical: exclusão mútua ampla; simples; degrada com alta contenção.
- atomic: incrementos isolados; menor overhead; impactado por false sharing/cache bouncing.

2. Ambiente de Teste

Item	Valor
CPU	AMD Ryzen 7 7700X
Núcleos físicos	8
Threads lógicas	16 (SMT)
Frequência Base/Boost	4.5 / 5.4 GHz
RAM	64 GB DDR5 5600 MHz
SO	Windows 11 Pro
Compilador	GCC 15.2.0
Flags	-O2 -Wall -fopenmp

3. Verificação de Corretude

Comandos PowerShell:

Ordenar resultados produzidos pelo analisador

```
Get-Content results_seq_dist.csv | Sort-Object | Set-Content  
results_seq_dist_sorted.csv
```

Ordenar gabarito

```
Get-Content gabarito_distribuido.csv | Sort-Object | Set-Content  
gabarito_distribuido_sorted.csv
```

Comparar (saída vazia = arquivos idênticos)

```
Compare-Object (Get-Content results_seq_dist_sorted.csv) (Get-Content
gabarito_distribuido_sorted.csv)
```

```
# Opcional: mensagem direta de OK/ERRO
```

```
if ( (Compare-Object (Get-Content results_seq_dist_sorted.csv) (Get-Content
gabarito_distribuido_sorted.csv)) ) {
```

```
    Write-Host 'DIFERENÇAS'
```

```
} else {
```

```
    Write-Host 'OK: resultados iguais ao gabarito'
```

```
}
```

```
Comando no MSYS2
```

```
# Ordenar (usa utilitário sort)
```

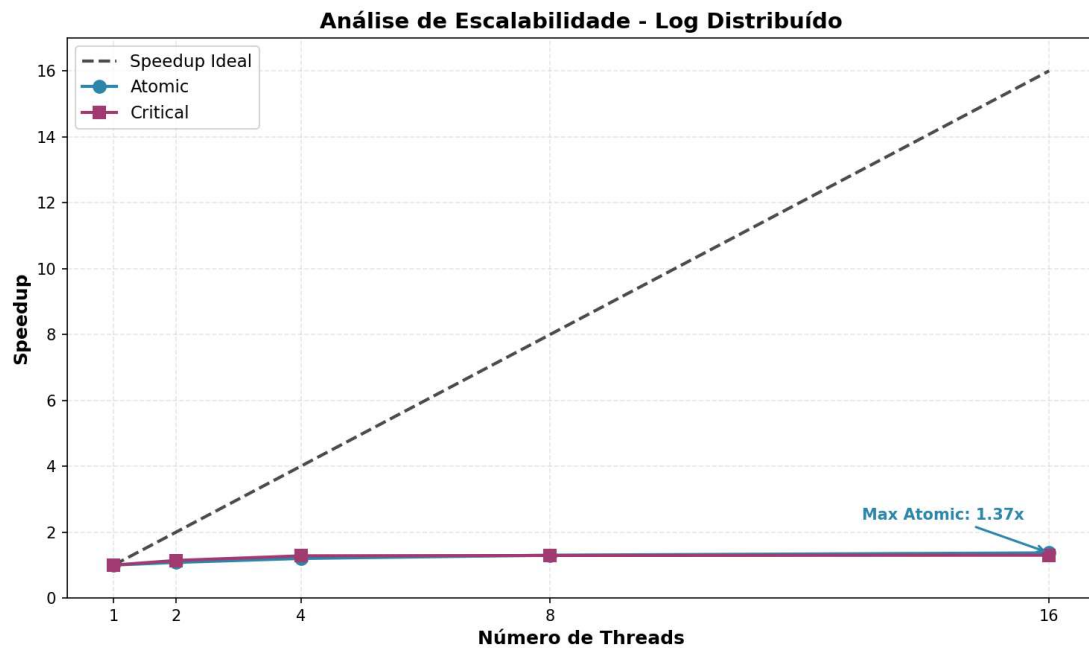
```
sort results_seq_dist.csv > results_seq_dist_sorted.csv
```

```
sort gabarito_distribuido.csv > gabarito_distribuido_sorted.csv
```

```
# Comparar silenciosamente
```

```
diff -q results_seq_dist_sorted.csv gabarito_distribuido_sorted.csv && \
```

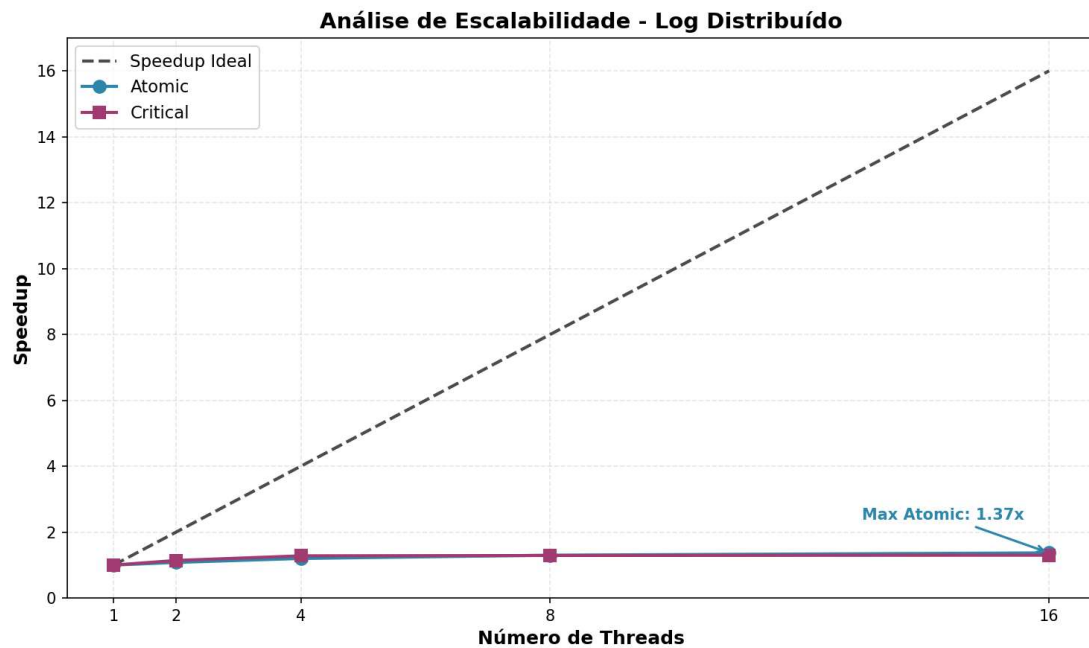
```
echo "OK: resultados iguais" || echo "DIFERENÇAS"
```

Resumo: Speedup máximo 1.37x (bem abaixo ideal). Ganhos decrescentes após 4 threads. Atomic e critical convergem; overhead de sincronização domina.

5. Resultados – Análise 2 (Contenção)

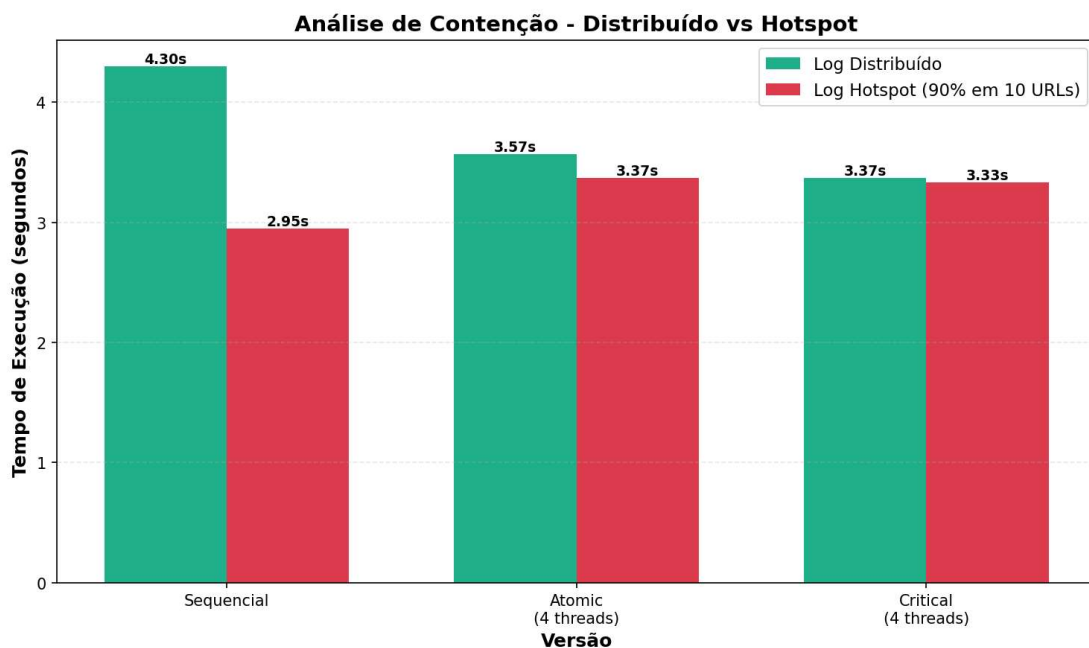
Versão	Log Distribuído (s)	Log Hotspot (s)	Diferença
Sequencial	4.30	2.95	-31.4%
par_atomic	3.57	3.37	-5.6%
par_critical	3.37	3.33	-1.2%



Resumo: Em hotspot a versão sequencial torna-se mais rápida. Contenção elimina ganhos paralelos.

6. Resultados – Análise 3 (Aplicação)

Top 10 URLs \approx 90% dos acessos (~900k cada de 10M).



Decisão de negócios: replicar imediatamente as 10 URLs (cache prioritário, nunca remover, pré-aquecimento, monitoramento em tempo real). Redução de carga na origem \approx 85–90%.

7. Conclusão e Análise

Escalabilidade

`par_critical` não escalou: serialização de cada incremento + lock contention.

`par_atomic` melhora levemente por granularidade fina, mas 10M operações atômicas + fração sequencial limitam speedup.

Contenção

Lock contention = múltiplas threads disputando a mesma variável/lock. Hotspot (90% em 10 itens) gera colisão constante; atomic sofre cache bouncing; critical cria fila.

Aplicação

Lentidão de `par_critical` atrasa detecção e replicação; em cenário real, decisão tardia perde o pico. Necessário abordagem com redução local ou streaming para latência baixa.

Síntese Final

Limites impostos por sincronização e concentração de acesso. Em carga uniforme há ganho modesto; em hotspot paralelismo bruto regrede. Melhorias: contadores locais + redução, sharding, batching, estruturas lock-free.