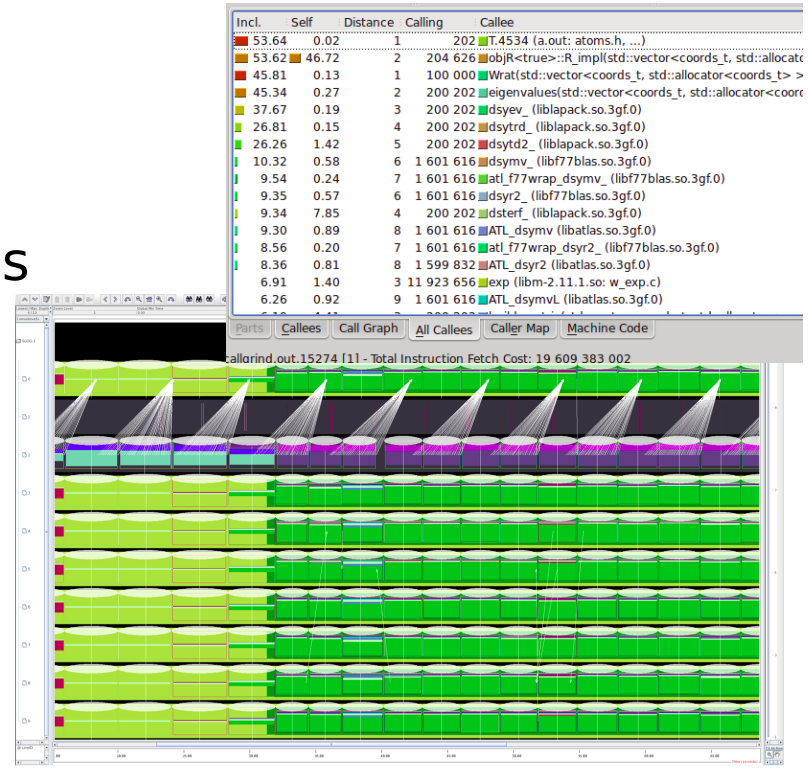
Profiling: técnicas e ferramentas

Profa. Andrea Charão
DLSC/CT/UFSM

Profiling

- Extração do perfil da execução de um programa
- Pode incluir, por exemplo:
  - Tempo gasto na execução do programa inteiro e de cada subprograma
  - Número de chamadas de funções ou métodos
  - Grafo de chamadas
  - Diagrama espaço-tempo (Gantt chart)
  - Etc.

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 52.39      0.67     0.67   160014     0.00     0.00  broadcast_msg(char*)
 17.99      0.90     0.23       48     4.80     9.38  catch_up_msg(int, int)
 17.20      1.12     0.22  4815505     0.00     0.00  send_msg(char*, int)
  7.04      1.21     0.09    10354     0.01     0.12  exec_cmds(SDL_Surface*)
  0.78      1.22     0.01   160062     0.00     0.00  parsemsg(char*)
  0.78      1.23     0.01   160014     0.00     0.00  formatandaddtomsgbuff(char*)
  0.78      1.24     0.01   160014     0.00     0.00  draw(int, int, int, int, int)
  0.78      1.25     0.01    10354     0.00     0.00  parse_cmds()
  0.78      1.26     0.01    10354     0.00     0.00  net_tick()
```



Como funciona o profiling

Técnica: Amostragem

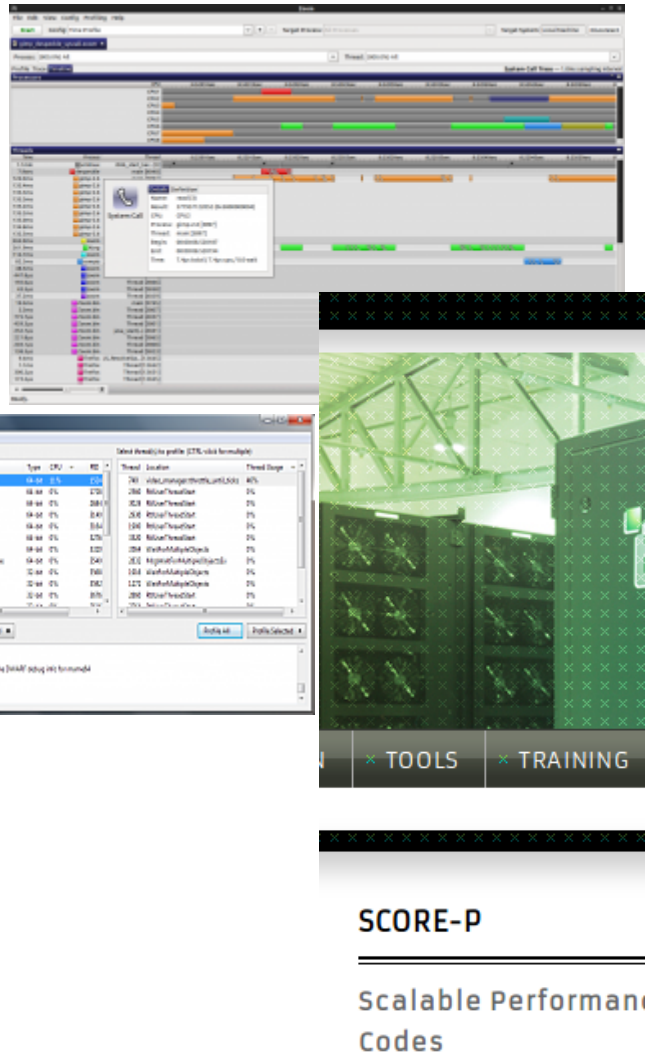
- interrupção periódica do programa para coleta de dados sobre a execução
- tende a ser pouco intrusiva
- menos precisa

Técnica: Instrumentação

- adição de instruções no programa para coleta de dados sobre a execução
- pode ser muito intrusiva
- mais precisa

Profilers

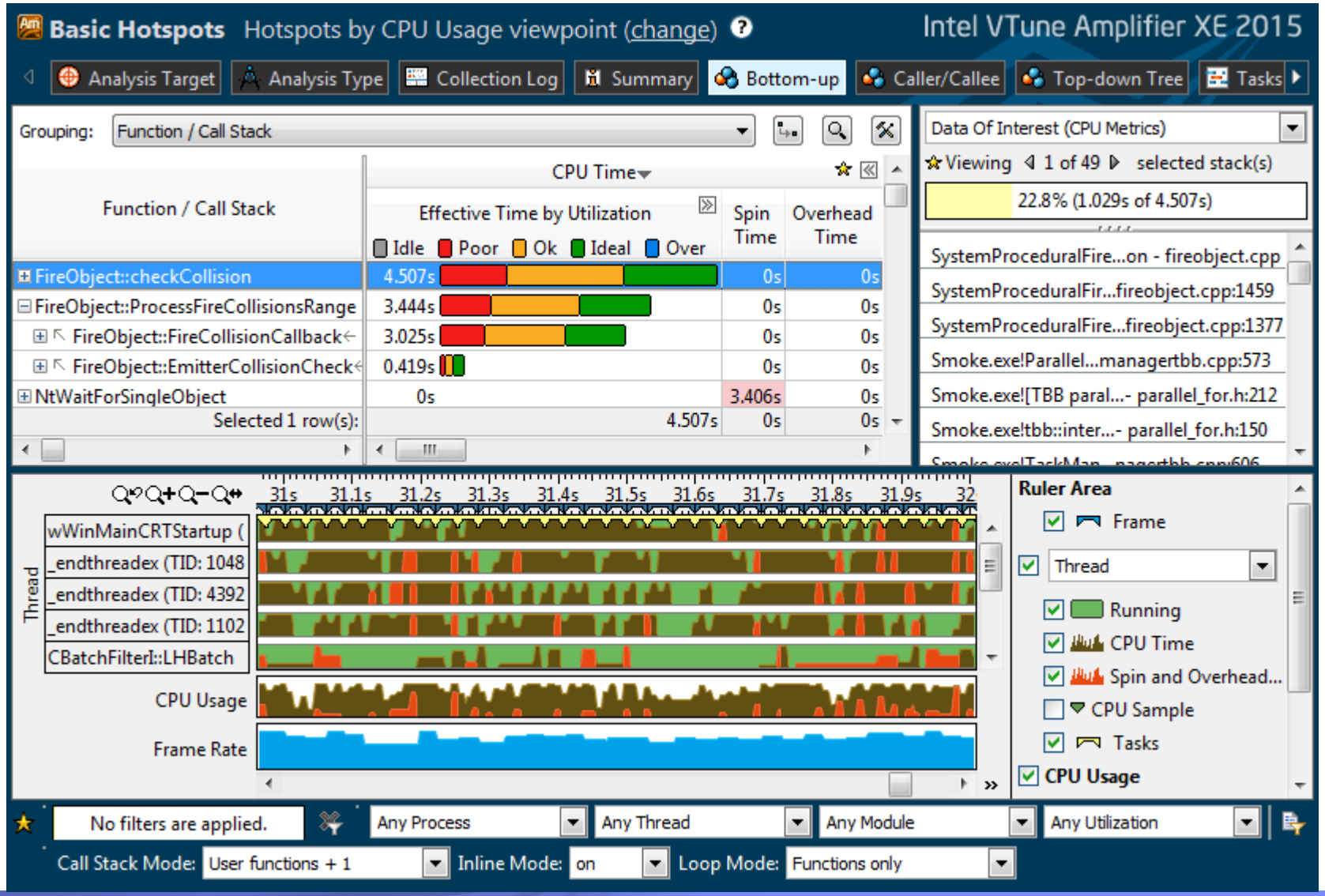
- Dependência (SO, compilador...)
- Análise “post-mortem” ou durante execução
- Suporte (ou não) a concorrência/paralelismo
- Exemplos:
  - Gprof, Kprof, Zoom, VerySleepy, Intel Vtune, etc.
  - Ferramentas avançadas: TAU, Score-P, PAPI, etc.



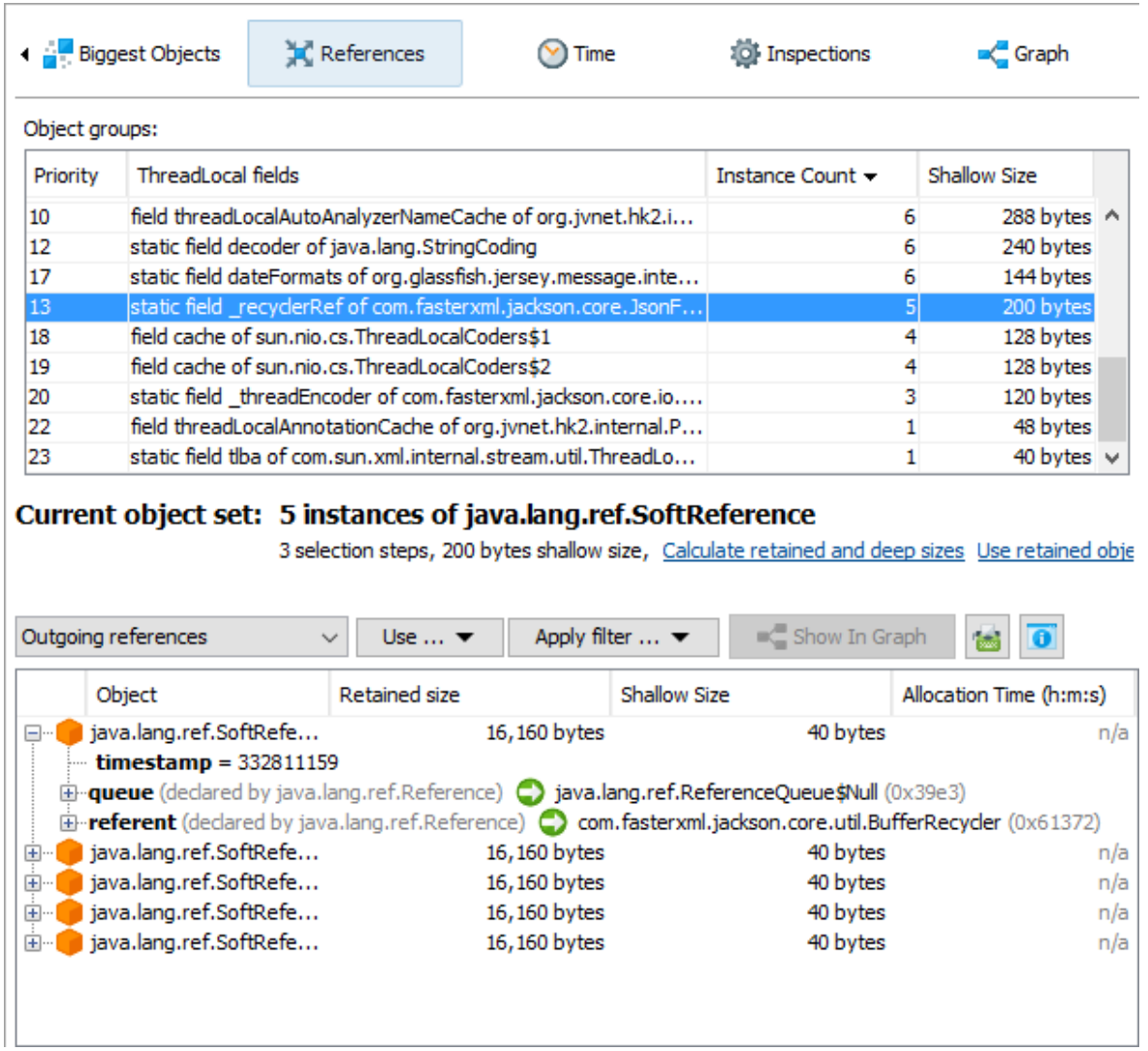
Exemplo: Gprof

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ms/call  ms/call  name
 52.39      0.67     0.67   160014     0.00     0.00  broadcast_msg(char*)
 17.99      0.90     0.23       48     4.80     9.38  catch_up_msg(int, int)
 17.20      1.12     0.22  4815505     0.00     0.00  send_msg(char*, int)
  7.04      1.21     0.09    10354     0.01     0.12  exec_cmds(SDL_Surface*)
  0.78      1.22     0.01   160062     0.00     0.00  parsemsg(char*)
  0.78      1.23     0.01   160014     0.00     0.00  formatandaddtomsgbuff(char*)
  0.78      1.24     0.01   160014     0.00     0.00  draw(int, int, int, int, int)
  0.78      1.25     0.01    10354     0.00     0.00  parse_cmds()
  0.78      1.26     0.01    10354     0.00     0.00  net_tick()
```

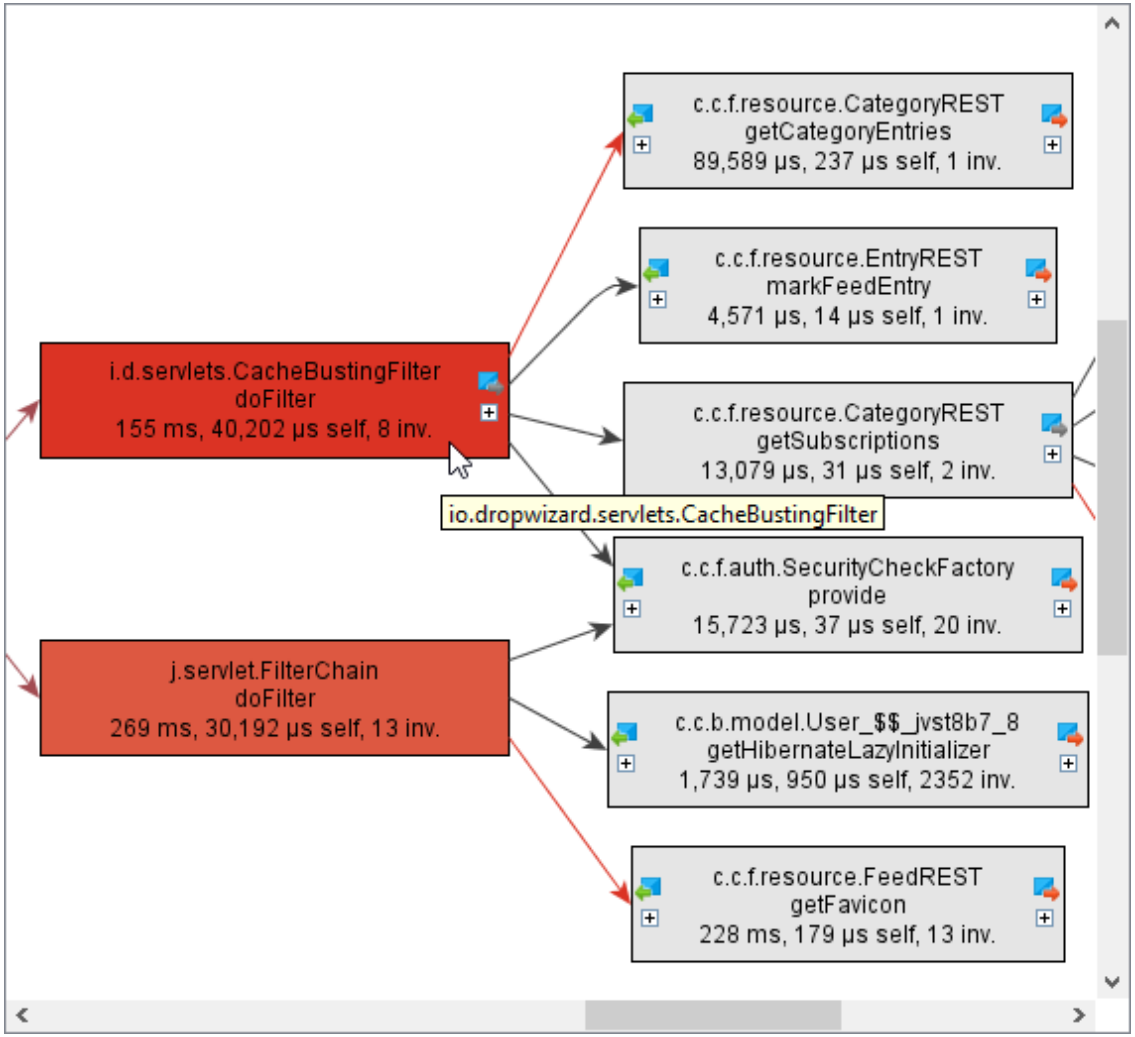
Exemplo: Intel Vtune Amplifier



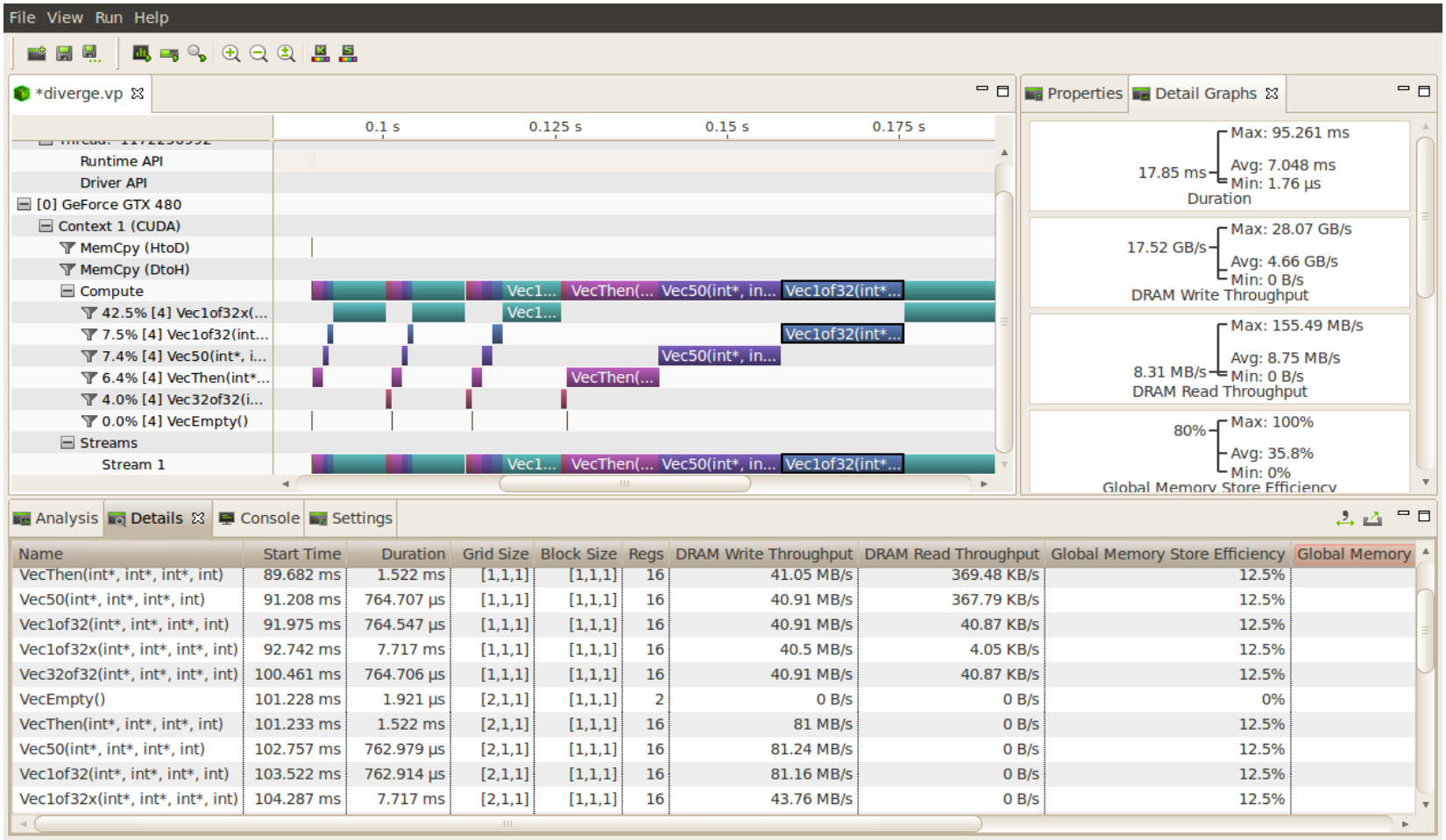
Exemplo: JProfiler



Exemplo: JProfiler



Exemplo: NVIDIA Visual Profiler



Name	Start Time	Duration	Grid Size	Block Size	Regs	DRAM Write Throughput	DRAM Read Throughput	Global Memory Store Efficiency	Global Memory
VecThen(int*, int*, int*, int)	89.682 ms	1.522 ms	[1,1,1]	[1,1,1]	16	41.05 MB/s	369.48 KB/s	12.5%	
Vec50(int*, int*, int*, int)	91.208 ms	764.707 µs	[1,1,1]	[1,1,1]	16	40.91 MB/s	367.79 KB/s	12.5%	
Vec1of32(int*, int*, int*, int)	91.975 ms	764.547 µs	[1,1,1]	[1,1,1]	16	40.91 MB/s	40.87 KB/s	12.5%	
Vec1of32x(int*, int*, int*, int)	92.742 ms	7.717 ms	[1,1,1]	[1,1,1]	16	40.5 MB/s	4.05 KB/s	12.5%	
Vec32of32(int*, int*, int*, int)	100.461 ms	764.706 µs	[1,1,1]	[1,1,1]	16	40.91 MB/s	40.87 KB/s	12.5%	
VecEmpty()	101.228 ms	1.921 µs	[2,1,1]	[1,1,1]	2	0 B/s	0 B/s	0%	
VecThen(int*, int*, int*, int)	101.233 ms	1.522 ms	[2,1,1]	[1,1,1]	16	81 MB/s	0 B/s	12.5%	
Vec50(int*, int*, int*, int)	102.757 ms	762.979 µs	[2,1,1]	[1,1,1]	16	81.24 MB/s	0 B/s	12.5%	
Vec1of32(int*, int*, int*, int)	103.522 ms	762.914 µs	[2,1,1]	[1,1,1]	16	81.16 MB/s	0 B/s	12.5%	
Vec1of32x(int*, int*, int*, int)	104.287 ms	7.717 ms	[2,1,1]	[1,1,1]	16	43.76 MB/s	0 B/s	12.5%	

Considerações finais

- Ferramentas avançadas de suporte ao desenvolvimento + otimização
- Um programa, diferentes profilers
  - Resultados podem variar