

# Alati za paralelno programiranje

- ❖ Adviser
- ❖ Composer
- ❖ Inspector
- ❖ Amplifier

# Parallel Advisor

## Pregled Advisor-ovih alata

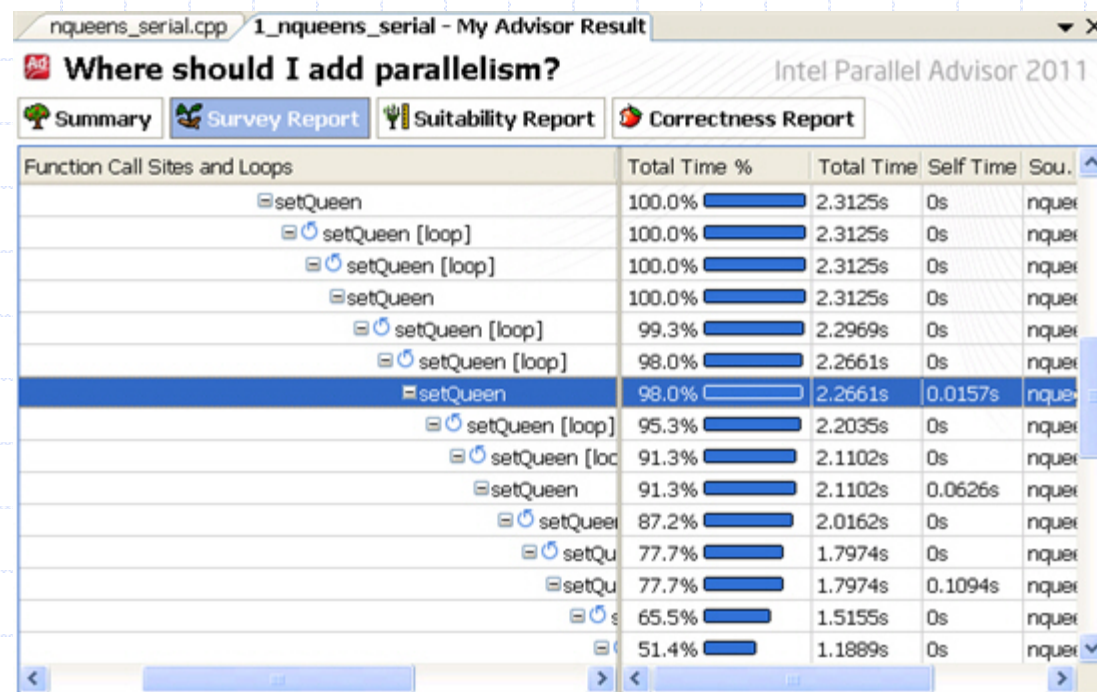
Alat i funkcija	Oznake
Survey. Otkriva mesta na kojima se troši najviše vremena. Performansa za određene linije koda.	Ne zahteva niti gleda oznake
VS editor. Unos oznaka. Nakon toga zamena oznaka stvarnim kodom paralelnog okruženja.	Omogućava unos i izmene oznaka
Suitability. Procena performanse za označene delove. Matematičko modeliranje performanse teorijski idealne paralelne mašine. Ovaj i prethodni alati zahteva Release konfiguraciju.	Zahteva oznake mesta i zadataka Prepoznaje neke oznake
Correctness. Prikazuje moguće probleme deljenja podataka, na osnovu oznaka paralelnih mesta, zadataka, ključeva, itd. Zahteva Debug konfiguraciju i ograničenje veličine podataka (x100 puta sporije izvršenje u odnosu na normalno).	Zahteva oznake mesta i zadataka Prepoznaje sve oznake



# Parallel Advisor

## Primer: nqueens\_Advisor – alat Survey (1/2)

- Projekat 1\_nqueens\_serial
  - ◆ Prevođenje u Release konfiguraciji
  - ◆ Izveštaj alata Survey
    - Najviše vremena nosi funkcija setQueen(), koja samu sebe rekurzivno poziva (pa se pojavljuje više puta u izveštaju)



# Parallel Advisor

## Primer: nqueens\_Advisor – alat Survey (2/2)

- Moguće je dobiti izveštaj na nivou izvornog koda
  - Total Time je procena vremena izvršenja iskaza ili funkcije koja se poziva iz iskaza
  - Loop Time je sumarno Total Time za sve osnovne blokove u petlji; prikazuje se jednom za celu petlju

The screenshot displays the Intel Parallel Advisor 2011 interface. The title bar indicates the file is 'nqueens\_serial.cpp' and the report is titled '1\_nqueens\_serial - My Advisor Result'. The main window shows a 'Where should I add parallelism? (Source)' report. The report includes tabs for 'Summary', 'Survey Report', 'Suitability Report', 'Correctness Report', and 'Survey Source'. The 'Survey Report' tab is active, showing a table with columns for 'Line', 'Source', 'Total Time', '%', 'Loop Time', and '%'. The table lists several code snippets, with the most significant one being the 'setQueen' function at line 88, which has a total time of 1.641s. The 'Call Stack with L...' panel on the right shows the call stack for the 'setQueen' function, including 'setQueen - nq...', 'solve - nqueen...', 'main - nqueen...', and '\_mainCRTSta...'. The 'Selected (Total Time):' at the bottom shows '0s'.

Line	Source	Total Time	%	Loop Time	%
77	// column is ok, set the queen				
78	queens[row]=col;				
79					
80	if(row==size-1) {	0.062s			
81	nrOfSolutions++; //Placed final queen				
82	}				
83	else {				
84	// try to fill next row				
85	for(int i=0; i<size; i++) {	0.016s			
86	//ADVISOR COMMENT: When surveying				
87	//ADVISOR COMMENT: Try looking at				
88	setQueen(queens, row+1, i);	1.641s			
89	}				
90	}				
91	}				
92					
93	/*				
94	Function to find all solutions for nQueens pro				

# Parallel Advisor

## Primer: nqueens\_Advisor – označavanje

- Analiza nakon pronalaženja vrućeg mesta setQueen()
  - ◆ Potrebno je posmatrati graf poziva od main do setQueen
    - Uočava se da je funkcija solve, koja poziva setQueen, zgodno mesto za uvođenje paralelizma – unesu se odgovarajuće oznake

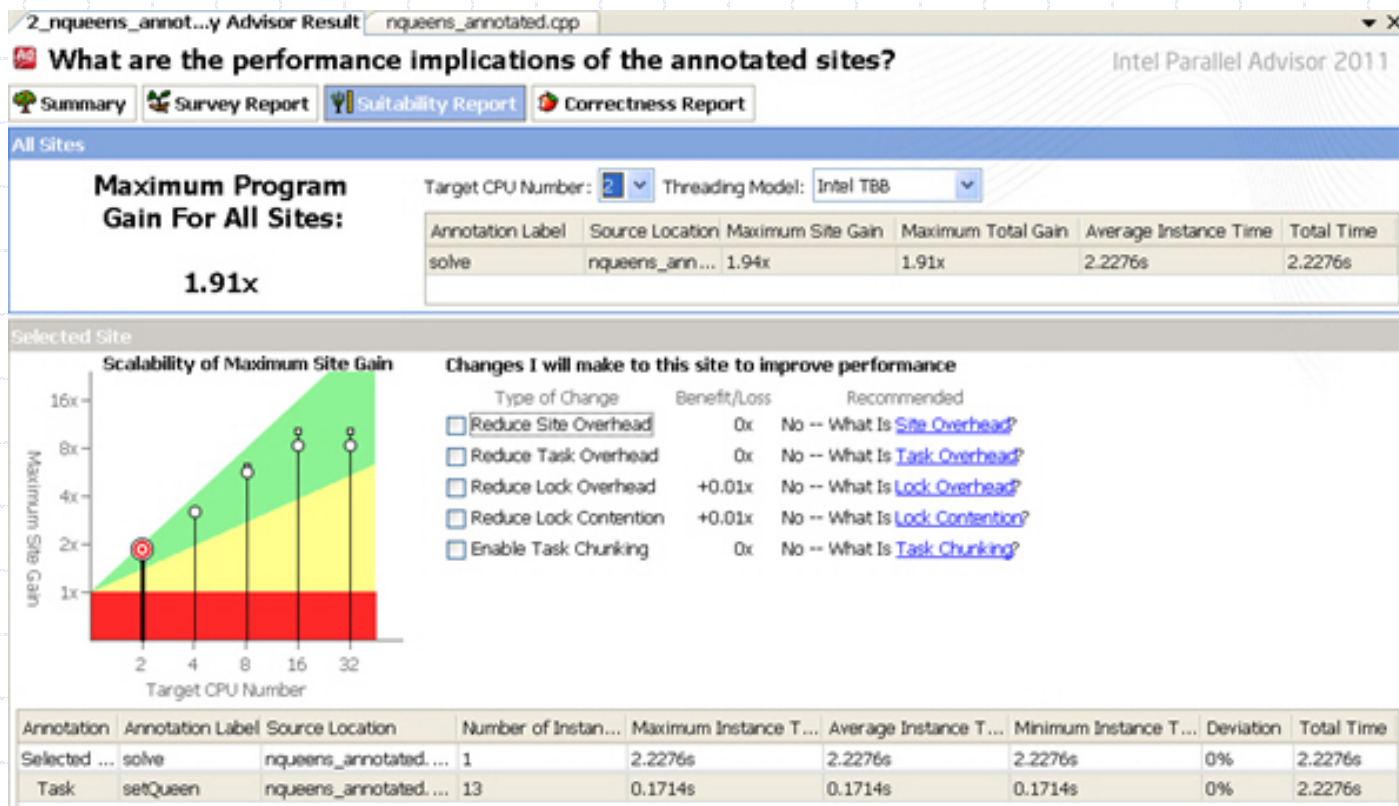
```
void solve() {  
    int * queens = new int[size];  
    ANNOTATE_SITE_BEGIN(solve)  
    for(int i=0; i<size; i++) {  
        ANNOTATE_TASK_BEGIN(setQueen)  
        setQueen(queens, 0, i);  
        ANNOTATE_TASK_END(setQueen)  
    }  
    ANNOTATE_SITE_END(solve)  
}
```



# Parallel Advisor

## Primer: nqueens\_Advisor – alat Suitability

- Projekat 2\_nqueens\_annotated
  - ◆ Prevođenje u Release konfiguraciji
  - ◆ Izveštaj alata Suitability – može se varirati br. jezgara



Cilj: Ostati u zelenoj oblasti skalabilnosti

Skalabilnost ograničena veličinom problema (ovde 13)

# Parallel Advisor

## Primer: nqueens\_Advisor – alat Correctness (1/2)

- Projekat 2\_nqueens\_annotated
  - ◆ Prevođenje u Debug konfiguraciji
  - ◆ Izveštaj alata Correctness – otkrivena 2 problema
    - Ponovno korišćenje memorije i komunikacija podataka

2\_nqueens\_annot...y Advisor Result nqueens\_annotated.cpp

Did the annotated tasks expose data sharing problems? Intel Parallel Advisor 2011

Summary Survey Report Suitability Report Correctness Report 1

Problems and Messages 2

ID	Problem	Sources	Modules	State
P1	Memory reuse	newaop.cpp; nquee ...	2_nqueens_ann ...	Not fixed
P2	Data communication	nqueens_annotated. ...	2_nqueens_ann ...	Not fixed
P3	Parallel site inform...	nqueens_annotated. ...	2_nqueens_ann ...	Not fixed

Filter 4

Severity

Error	2
Remark	1

Problem

Data communication	1
Memory reuse	1
Parallel site informat...	1

Source

newaop.cpp	1
nqueens_annotated. ...	3

Module

2_nqueens_annotat...	3
----------------------	---

State

Not fixed	3
-----------	---

Memory reuse: Observations 3

ID	Description	Source	Function	Module	State
X3	Allocation ...	newaop.cpp:7	new[]	2_nqueens_annotat ...	Informa...
X4	Parallel site	nqueens_annotated ...	solve	2_nqueens_annotat ...	Informa...
X5	Write	nqueens_annotated ...	setQueen	2_nqueens_annotat ...	Not fi...
X6	Write	nqueens_annotated ...	setQueen	2_nqueens_annotat ...	Not fi...
X7	Read	nqueens_annotated ...	setQueen	2_nqueens_annotat ...	Not fi...

```
69     for(int i=0; i<row; i++) {
70         // vertical attacks
71         if (is queens[i] == col) {
```

Sort By Item Name



# Parallel Advisor

## Primer: nqueens\_Advisor – alat Correctness (2/2)

- Dvostruki klik na problem ponovnog korišćenja mem.
  - ◆ Otvara se fokus prozor i prozor koji je povezan sa njim
  - ◆ Dijagram odnosa: dodela-upis-upis
    - Nakon dodele niza queens, paralelan upis u njega iz 2 zadatka

The screenshot displays the Intel Parallel Advisor 2011 interface. The main window title is "2\_nqueens\_annot...y Advisor Result nqueens\_annotated.cpp". The status bar indicates "Did the annotated tasks expose data sharing problems? (Source)". The interface includes tabs for "Summary", "Survey Report", "Suitability Report", "Correctness Report", and "Correctness Source".

The "Focus Observation" pane shows a write operation at line 81 of nqueens\_annotated.cpp:81 - Write. The code snippet is as follows:

```
80 //ADVISOR COMMENT: See comment at top of function
81 queens[row]=col;
82
83 if(row==size-1) {
```

The "Related Observation" pane shows a similar write operation at line 81 of nqueens\_annotated.cpp:81 - Write. The code snippet is as follows:

```
80 //ADVISOR COMMENT: See comment at top of function
81 queens[row]=col;
82
83 if(row==size-1) {
```

The "Call Stack" pane shows the following stack:

- setQueen - nqueens\_annotated.cpp:81
- solve - nqueens\_annotated.cpp:120
- main - nqueens\_annotated.cpp:157

The "Memory reuse: Observations" table shows the following data:

ID	Descrip...	Source	Funct...	Module	State
X3	Allocati...	newaop.cpp:7	new[]	2_nqueens_ann...	Infor...
X4	Parallel...	nqueens_ann...	solve	2_nqueens_ann...	Infor...
X5	Write	nqueens_ann...	setQu...	2_nqueens_ann...	Not...
X6	Write	nqueens_ann...	setQu...	2_nqueens_ann...	Not...
X7	Read	nqueens_ann...	setQu...	2_nqueens_ann...	Not...

The "Relationship Diagram" pane shows a flow from "Allocation site" (newaop.cpp:7) to "Write" (nqueens\_anno) and then to "Write" (nqueens\_annot).

# Parallel Advisor

## Primer: nqueens\_Advisor – sumarni izveštaj

- Dobija se nakon upotrebe Suitability i Correctness
  - ◆ Prikazuje moguće dobitke u performansi zajedno sa problemima, koji se pojavljuju zbog paralelizacije
    - Vredi nastaviti jer je Maximum Self Gain dobar,
    - a pojavljuju se samo 2 problema

2\_nqueens\_annot...y Advisor Result nqueens\_annotated.cpp

**Summary of predicted parallel behavior** Intel Parallel Advisor 2011

Summary Survey Report Suitability Report Correctness Report

Annotation	Source Location	Annotation Label	Self Time	Maximum Self Gain	Maximum Total Gain	Correctness Problems
Site	nqueens_annotated.cpp:113	solve	2.2276s	1.94x	1.91x	2 errors 0 warnings
Site End	nqueens_annotated.cpp:123	solve	-	-	-	- -
Task	nqueens_annotated.cpp:117	setQueen	2.2276s	-	-	- -
Task End	nqueens_annotated.cpp:121	setQueen	-	-	-	- -
Annotation	nqueens_annotated.cpp:47	advisor-annotate.h	-	-	-	- -

Modeling Assumptions: Target CPU Number: 2; Threading Model: Intel TBB

# Parallel Advisor

## Primer: nqueens\_Advisor – rešenje problema (1/2)

- Problem ponovnog korišćenja memorije
  - ◆ Komentariše se originalna dodela niza queens, pre petlje
  - ◆ Ta dodela se uvodi (praktično prebacuje) u telo petlje
    - Tako svaki zadatak ima svoju instancu niza

```
void solve() {  
    //int * queens = new int[size];    //original allocation  
    ANNOTATE_SITE_BEGIN(solve)  
    for(int i=0; i<size; i++) {  
        ANNOTATE_TASK_BEGIN(setQueen)  
        // create separate array for each recursion  
        int * queens = new int[size];  
        setQueen(queens, 0, i);  
        ANNOTATE_TASK_END(setQueen)  
    }  
    ANNOTATE_SITE_END(solve)  
}
```

# Parallel Advisor

## Primer: nqueens\_Advisor – rešenje problema (2/2)

- Problem komunikacije podataka
  - ♦ Izaziva ga paralelno izvršenje iskaza: `nrOfSolutions++`;
    - Nekoordiniran upis iz paralelnih zadataka u tu promenljivu
  - ♦ Rešenje: uvesti zaključavanje te promenljive (lock)

```
queens[row]=col;
if(row==size-1) {
    ANNOTATE_LOCK_ACQUIRE(0)
    nrOfSolutions++; //Placed final queen, found a solution!
    ANNOTATE_LOCK_RELEASE(0)
}
else {
    // try to fill next row
    for(int i=0; i<size; i++) {
        setQueen(queens, row+1, i);
    }
}
```

# Parallel Advisor

## Primer: nqueens\_Advisor –paralelizacija koda (1/2)

- Oznake se zamenjuju stvarnim kodom okruženja
  - ◆ Intel Cilk
    - Projekat 3\_nqueens\_cilk
    - Dodate #include direktive, for u funkciji solve zamenjen sa cilk\_for, problem povećanja nrOfSolutions rešen uvođenjem Cilk reduktora za promenljivu nrOfSolutions
      - `cilk::reducer_opadd<int> nrOfSolutions;`
  - ◆ Intel TBB
    - Projekat 3\_nqueens\_tbb
    - Dodate #include direktive, parallel\_for u funkciji solve, oznake zaključavanja zamenjene odgovarajućim TBB kodom – naredni slajd



# Parallel Advisor

## Primer: nqueens\_Advisor –paralelizacija koda (2/2)

```
{  
  // increment is not atomic, so setting a lock is required here  
  NrOfSolutionsMutexType::scoped_lock mylock(NrOfSolutionsMutex);  
  nrOfSolutions++; //Placed final queen, found a solution!  
} // closing } invokes scoped_lock destructor which releases the lock
```

# Parallel Composer

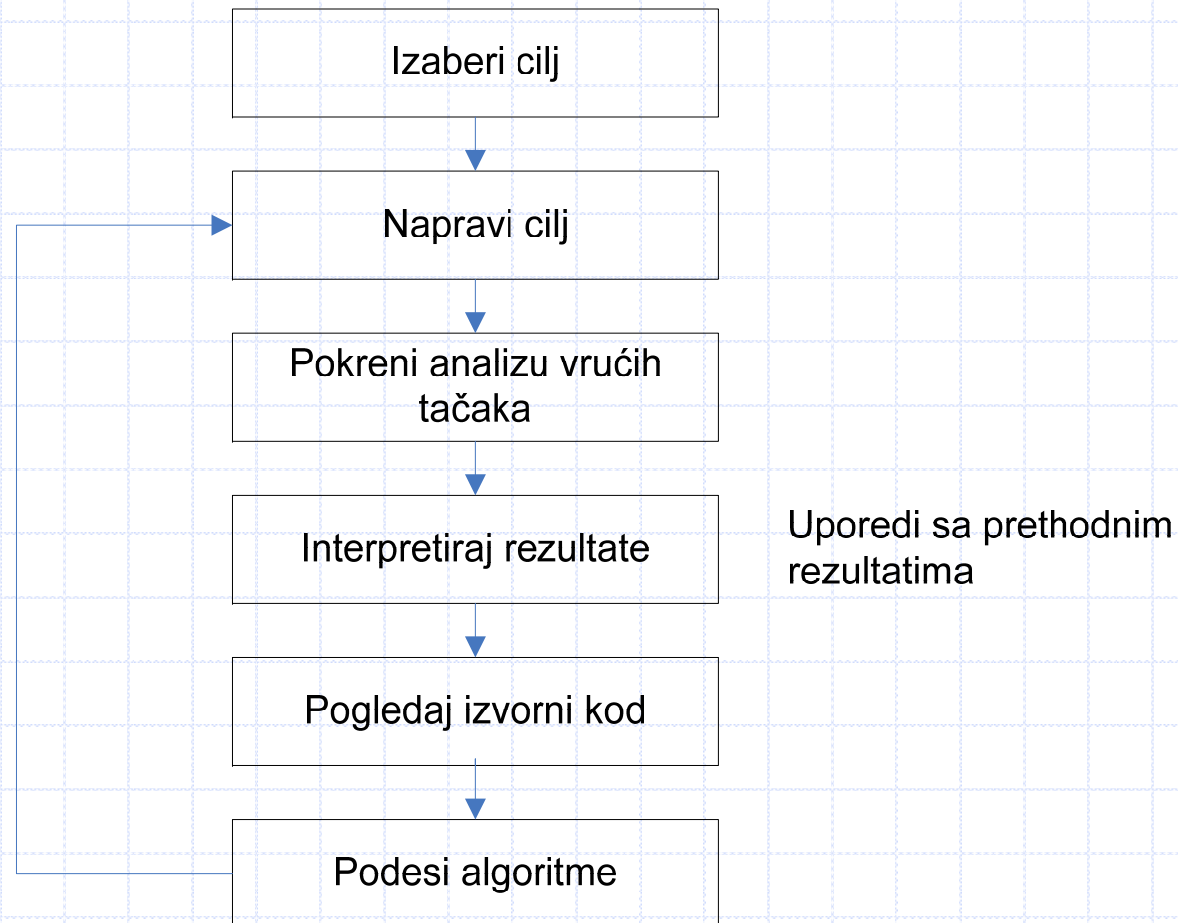
## ◆ Cine ga

- Intel® C++ Compiler
- Intel® Cilk™ Plus
- Intel® Integrated Performance Primitives
- Intel® Threading Building Blocks
- Intel® Parallel Debugger Extension

# Parallel Amplifier

- ◆ Obezbeđuje info o performansi programa:
  - Identifikovanje vrućih tačaka (hot spots)
  - Lociranje sekcija koda koje ne iskorišćuju raspoloživo vreme (na svim jezgrima)
  - Određivanje najbolje sekcije koda za optimizaciju
  - Lociranje sinhronizacionih objekata koji utiču na performansu aplikacije
  - Pronalaženje da li, gde, i zašto, aplikacija troši vreme na U-I radnje
  - Analiza performanse različitih sinhronizacionih metoda, različitog broja niti, ili različitih algoritama

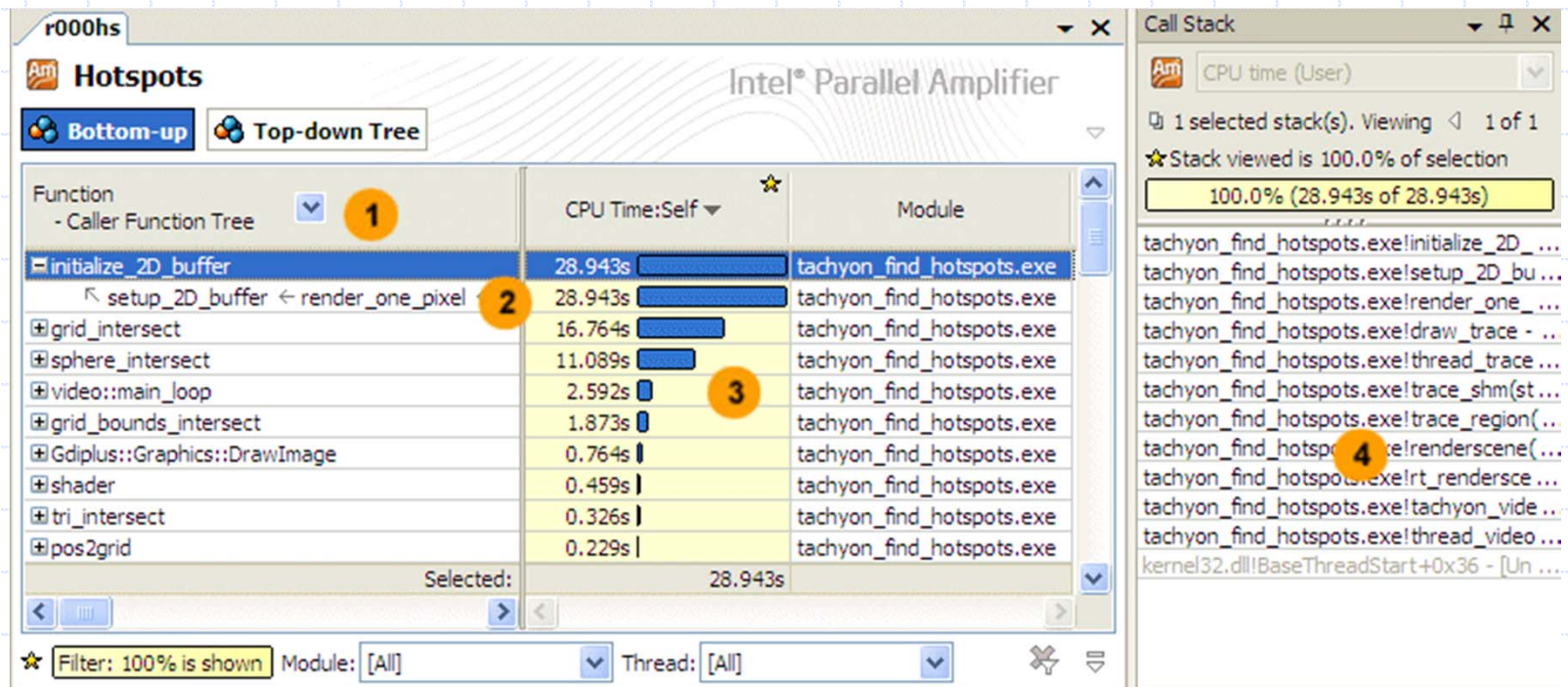
# Parallel Amplifier: Pronalaženje vrućih tačaka



# Primer: find\_hotspots

1. Opcija: Where is my program spending time?

◆ 3: Vidi se da je vruća tačka funkcija initialize\_2D\_buffer – ona troši najviše vremena



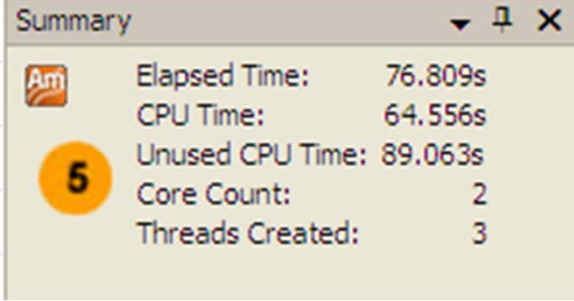


# Primer: find\_hotspots

## 2. Analiza iskorišćenja procesora

### ◆ Metrike

- Elapsed time – proteklo vreme
- CPU time – ukupno iskorišćeno vreme za sve niti
- Unused CPU time – ukupno neiskorišćeno vreme za sva jezgra
- Core Count – broj jezgara (logičkih CPU)
- Threads Created – broj izvršenih niti



A screenshot of a 'Summary' window with a title bar containing a dropdown arrow, a maximize icon, and a close icon. The window displays performance metrics for an application, with an 'Am' icon and a yellow circle containing the number '5' on the left. The metrics are listed in a two-column format.

Elapsed Time:	76.809s
CPU Time:	64.556s
Unused CPU Time:	89.063s
Core Count:	2
Threads Created:	3

# Primer: find\_hotspots

## 3. Analiza na nivou iskaza u funkciji

Intel® Parallel Amplifier

Hotspots

Bottom-up Top-down Tree find\_hotspots.cpp

4

Line	Source	CPU Time:Self
78	{	
79	// First (slower) method of filling array	
80	// Array is NOT filled in consecutive memory address order	
81	/*****	
82	for (int i = 0; i < mem_array_i_max; i++)	
83	{	
84	for (int j = 0; j < mem_array_j_max; j++)	0.110s
85	{	
86	mem_array[j*mem_array_j_max+i] = *fill_value + 2;	28.044s
87	}	
88	}	
89		
90		
91	/*****	
92		
93		
94	// Faster method of filling array	
95	// The for loops are interchanged	
96	// Array IS filled in consecutive memory address order	
97	/*****	
98	for (int j = 0; j < mem_array_j_max; j++)	
99	{	
100	for (int i = 0; i < mem_array_i_max; i++)	
101	{	
102	mem_array[j*mem_array_j_max+i] = *fill_value + 2;	
103	}	
104	}	
105	/*****	

Selected (1 row(s)): 28.044s

Filter: 100% is shown Module: [All] Thread: [All] Process: [All]

# Primer: find\_hotspots

## 4. Uočavanje problema

- ◆ Najviše vremena troši inicijalizacija niza, koja se ne obavlja u redosledu susednih mem. adresa
- ◆ Rešenje – preraditi inicijalizaciju niza tako da se obavlja u redosledu susednih mem. adresa
- ◆ Ovde se to efektivno postiže zamenom redosleda ugnježđenih petlji
- ◆  $\text{for}(1) \{ \text{for}(2) \{ \dots \} \} \Rightarrow \text{for}(2) \{ \text{for}(1) \{ \dots \} \}$

# Primer: find\_hotspots

## 5. Poređenje prethodnog i novog rešenja

### ◆ I Analiza na nivou pojedinačnih funkcija

Hotspots

Bottom-up

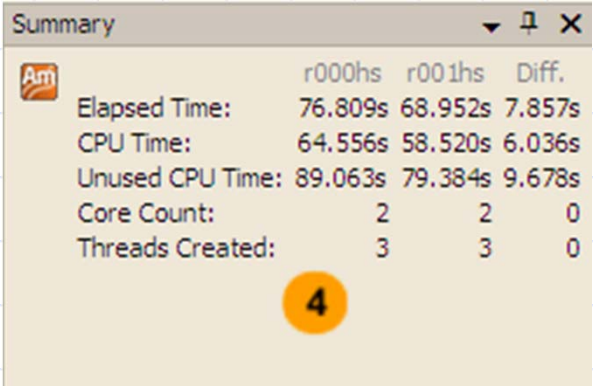
Function	CPU Time:Self: r000hs	CPU Time:Self: r001hs	CPU Time:Self:Difference	Module
initialize_2D_buffer	28.943s	22.427s	6.517s	tachyon_find_hotspots.exe
grid_intersect	16.764s	15.352s	1.412s	tachyon_find_hotspots.exe
Gdiplus::Graphics::DrawImage	0.764s	0.218s	0.546s	tachyon_find_hotspots.exe
intersect_objects	0.172s	0.064s	0.108s	tachyon_find_hotspots.exe
ColorScale	0.157s	0.091s	0.066s	tachyon_find_hotspots.exe
GetString	0.062s	0s	0.062s	tachyon_find_hotspots.exe
light_intersect	0.140s	0.086s	0.054s	tachyon_find_hotspots.exe
shader	0.459s	0.410s	0.049s	tachyon_find_hotspots.exe
shade_reflection	0.031s	0.011s	0.021s	tachyon_find_hotspots.exe
_SEH_prolog4	0.016s	0s	0.016s	MSVCR80.dll
WinInit	0.016s	0s	0.016s	tachyon_find_hotspots.exe
Selected:	28.943s	22.427s	6.517s	

Filter: 100% is shown Module: [All] Thread: [All] Process: [All]

# Primer: find\_hotspots

## 5. Poređenje prethodnog i novog rešenja

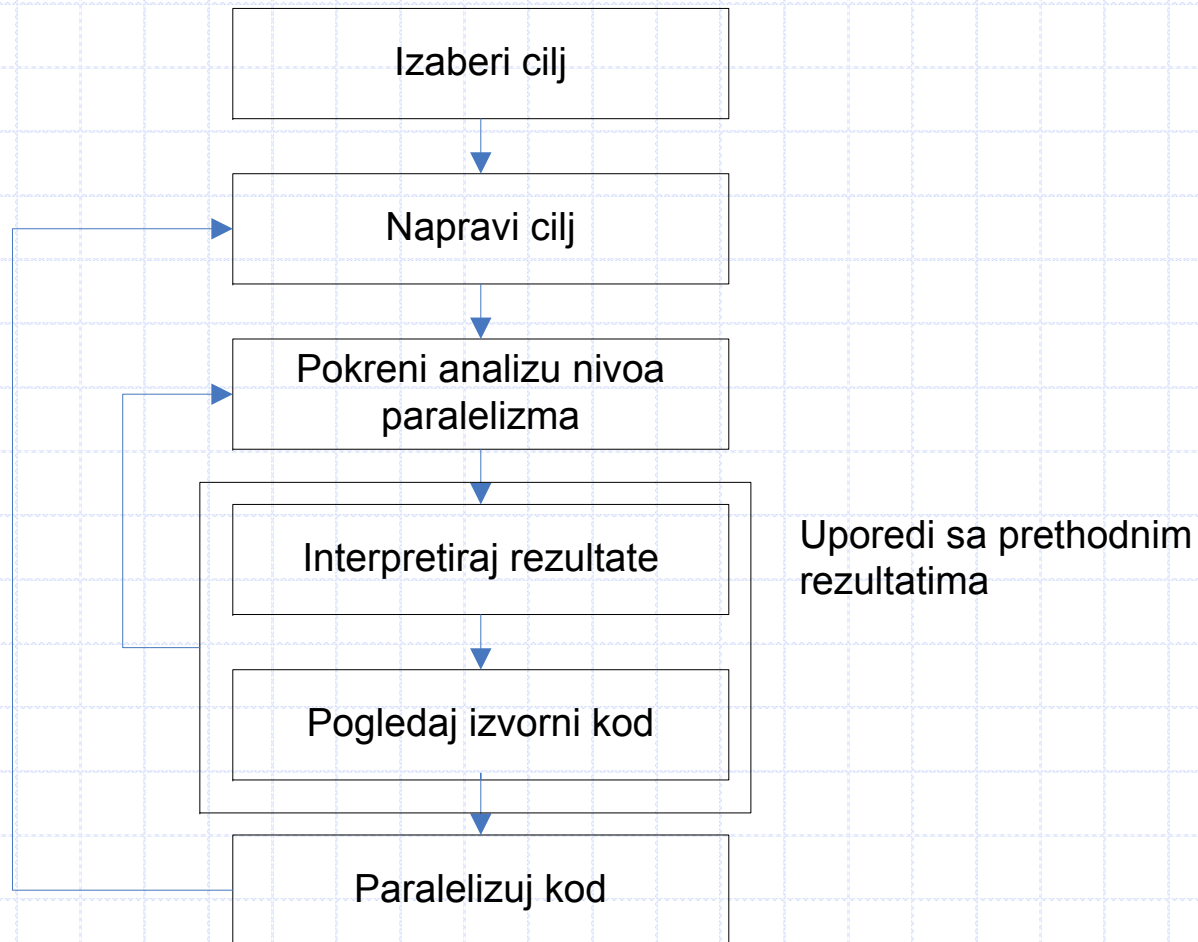
- ◆ II Poređenje na nivou iskorišćenja ukupnog vremena
- ◆ Iskorišćenje procesora u prvom slučaju iznosi 42.02%, a u drugom 42.43%
- ◆ Ubrzanje je 1.14



	r000hs	r001hs	Diff.
Elapsed Time:	76.809s	68.952s	7.857s
CPU Time:	64.556s	58.520s	6.036s
Unused CPU Time:	89.063s	79.384s	9.678s
Core Count:	2	2	0
Threads Created:	3	3	0



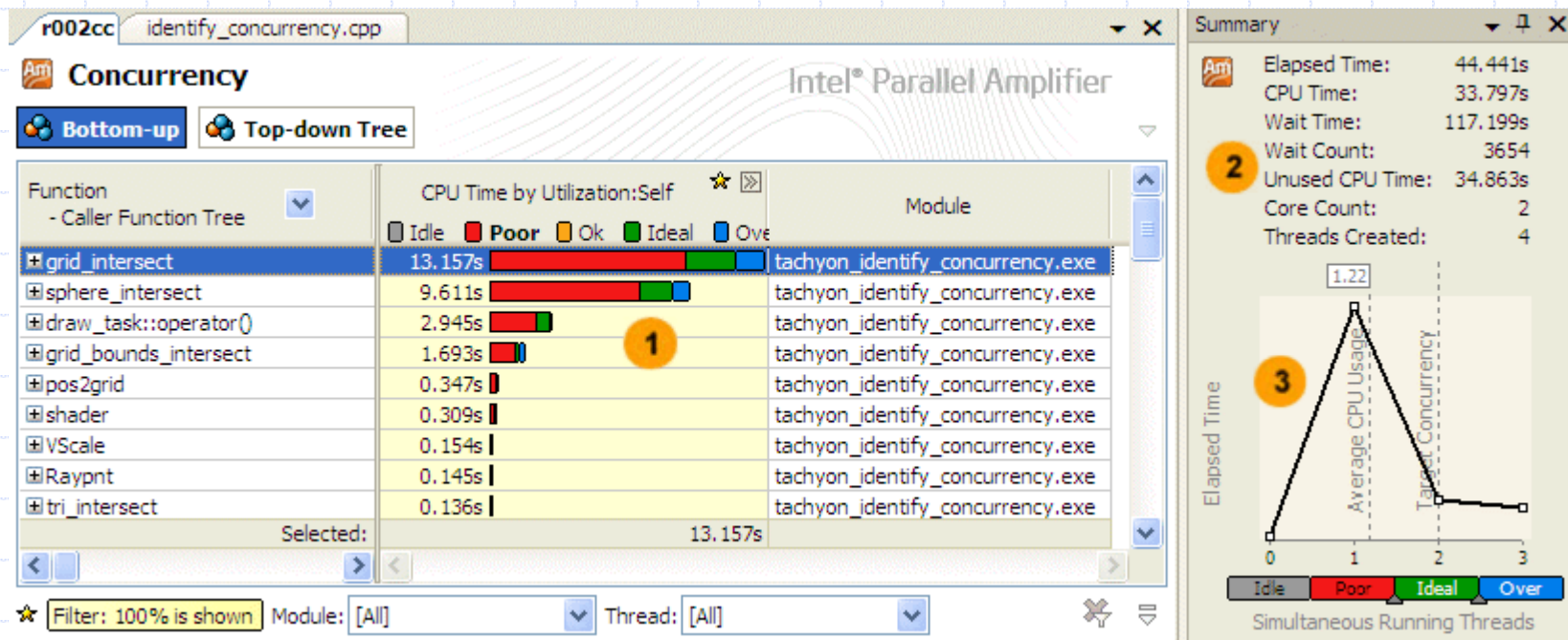
# Parallel Amplifier: Povećanje nivoa paralelizma



# Primer: identify\_concurrency

## 1. Opcija: Where is my concurrency poor?

- ◆ 1: Najlošiji paralelizam je u funkciji grid\_intersect



# Primer: identify\_concurrency

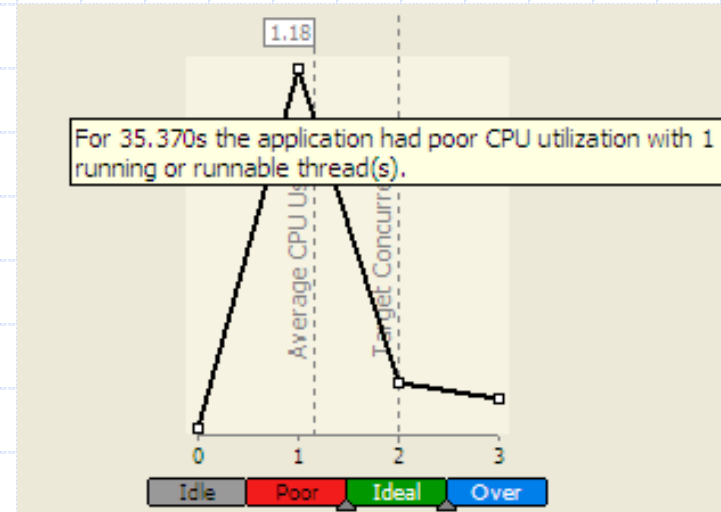
## 1a. Definicije metrika

- ◆ Ciljni paralelizam (Target Concurrency) je jednak broju jezgara u CPU
- ◆ Npr. Za CPU sa 2 jezgra,  $TC=2$
- ◆ Prosečna upotreba procesora  $AU = T_{cpu} / T_e$ 
  - Gde su
    - ◆ AU – prosečna upotreba CPU (Average CPU usage)
    - ◆  $T_{cpu}$  – vreme rada svih jezgara (CPU time)
    - ◆  $T_e$  – ukupno proteklo vreme (Elapsed Time)
- ◆ AU idealno treba da teži broju jezgara

# Primer: identify\_concurrency

## 1b. Analiza AU i tragova izvršenja programa

◆ Sa grafika se vidi da se jedna nit izvršavala 35s, što se klasifikuje kao loš (poor) paralelizam



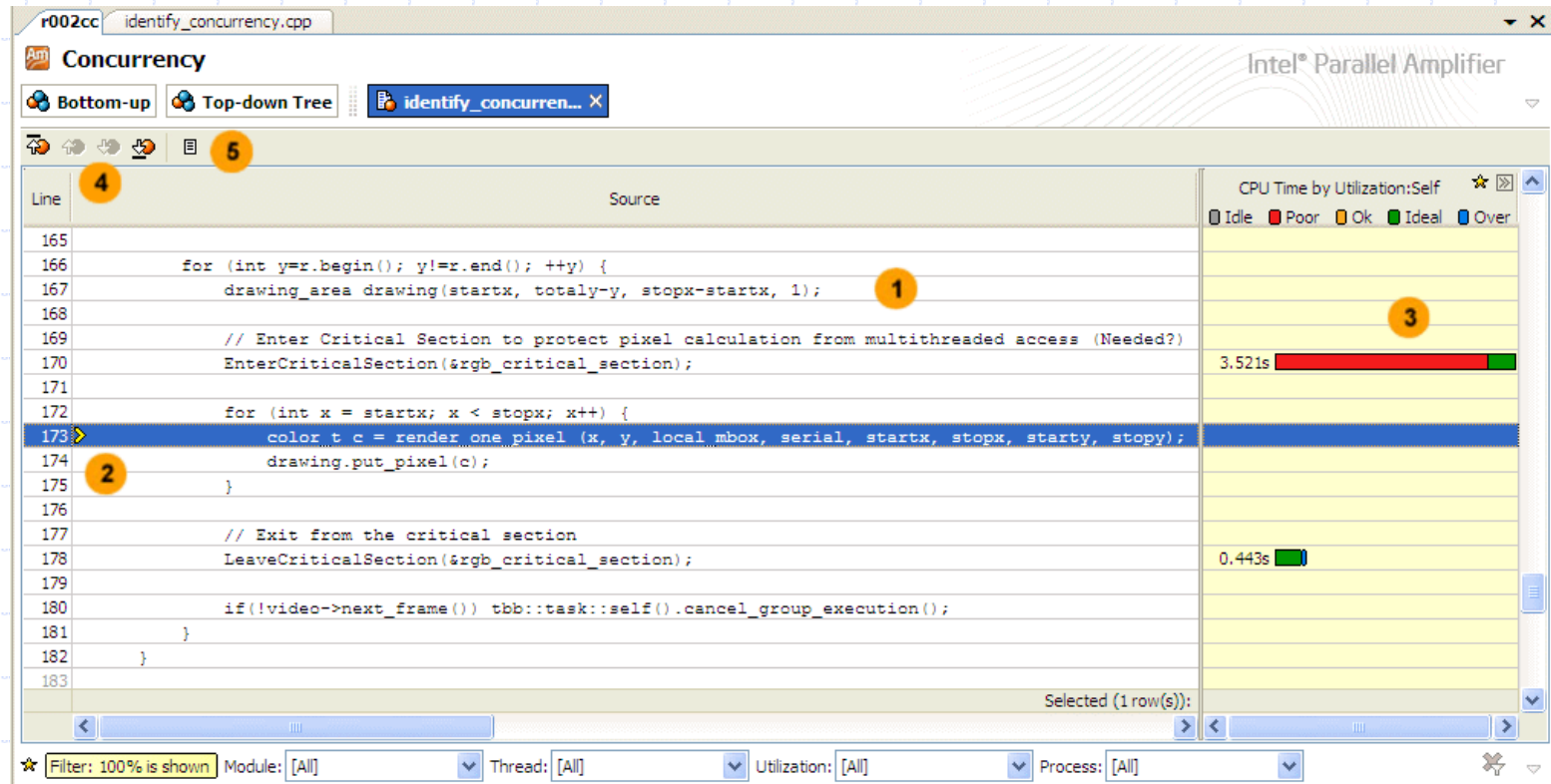
◆ Otvaranjem steka poziva, vide se tri traga izvršenja

grid_intersect	14.655s	<div><div></div><div></div><div></div><div></div></div>
+ shader ← trace	8.988s	<div><div></div><div></div><div></div><div></div></div>
+ trace	4.827s	<div><div></div><div></div><div></div><div></div></div>
+ grid_intersect	0.840s	<div><div></div><div></div><div></div><div></div></div>

# Primer: identify\_concurrency

## 2. Analiza na nivou iskaza

- Analizom steka poziva funkcija, uočava se da je funkcija `draw_task` uzrok lošeg paralelizma

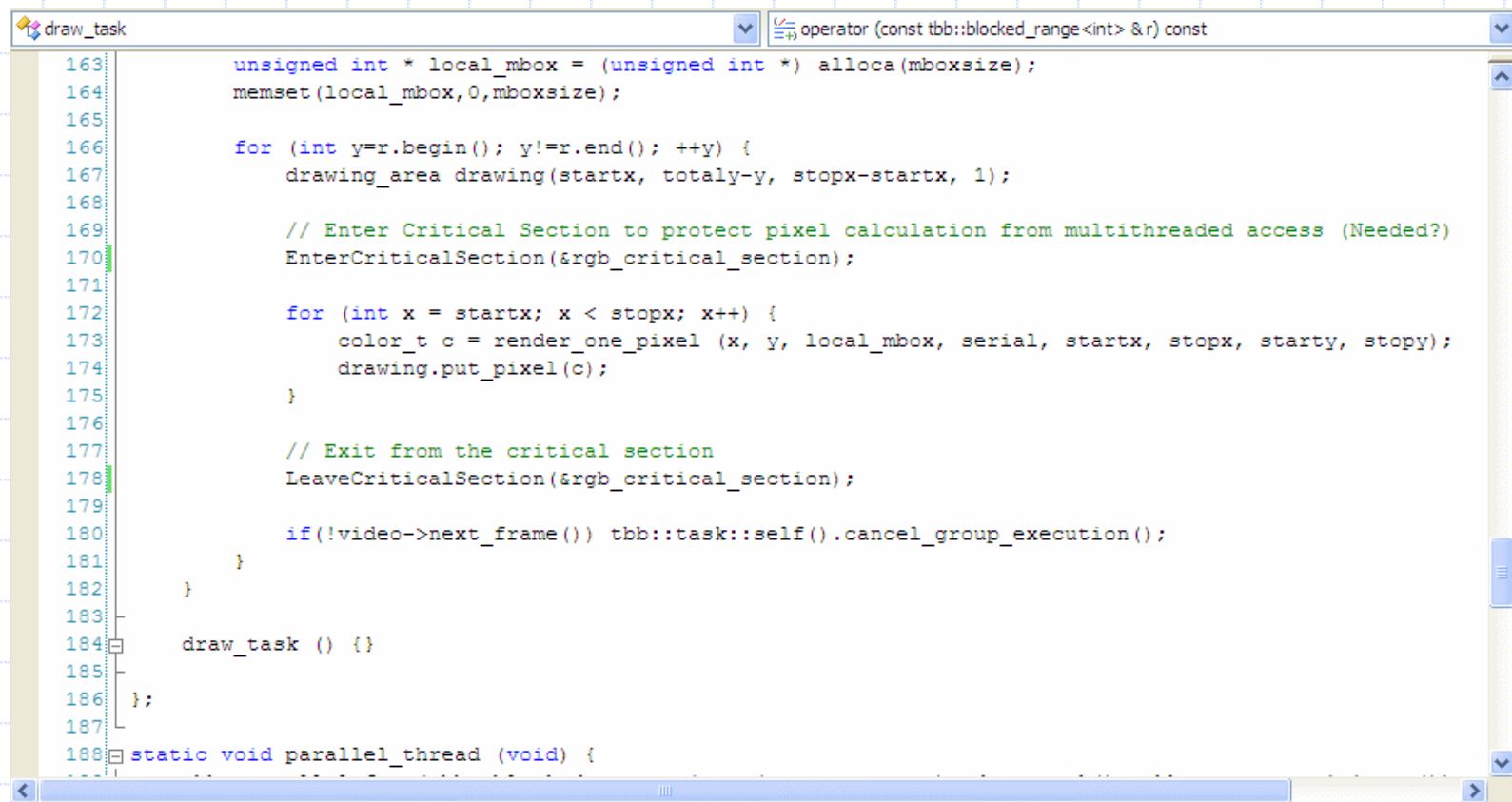




# Primer: identifiy\_concurrency

## 3. Uočavanje problema i rešenje

◆ Rešenje: uklanjanje nepotrebne kritične sekcije

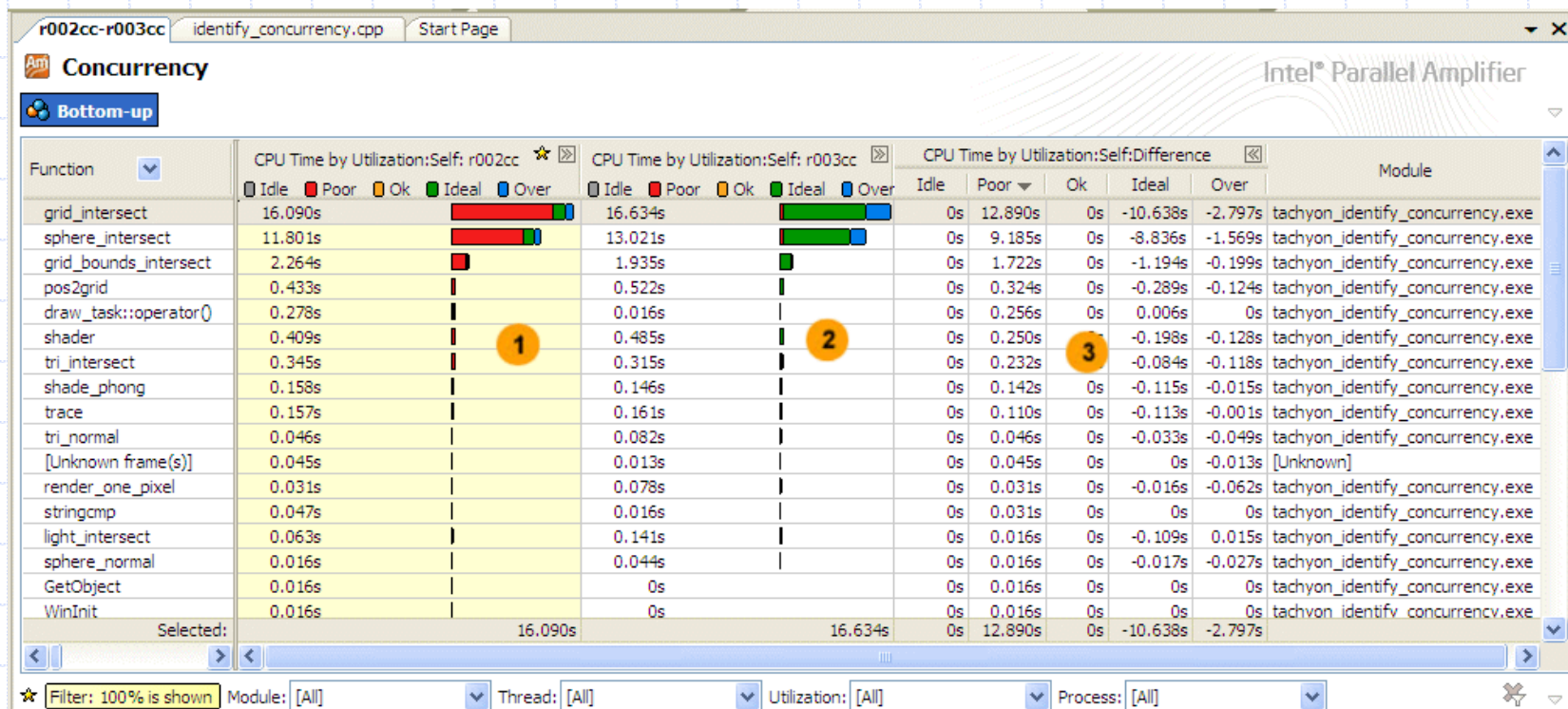


```
163 unsigned int * local_mbox = (unsigned int *) alloca(mboxsize);
164 memset(local_mbox, 0, mboxsize);
165
166 for (int y=r.begin(); y!=r.end(); ++y) {
167     drawing_area drawing(startx, totaly-y, stopx-startx, 1);
168
169     // Enter Critical Section to protect pixel calculation from multithreaded access (Needed?)
170     EnterCriticalSection(&rgb_critical_section);
171
172     for (int x = startx; x < stopx; x++) {
173         color_t c = render_one_pixel (x, y, local_mbox, serial, startx, stopx, starty, stopy);
174         drawing.put_pixel(c);
175     }
176
177     // Exit from the critical section
178     LeaveCriticalSection(&rgb_critical_section);
179
180     if(!video->next_frame()) tbb::task::self().cancel_group_execution();
181 }
182
183
184 draw_task () {}
185
186 };
187
188 static void parallel_thread (void) {
```

# Primer: identify\_concurrency

## 4. Upoređenje sa prethodnim rešenjem

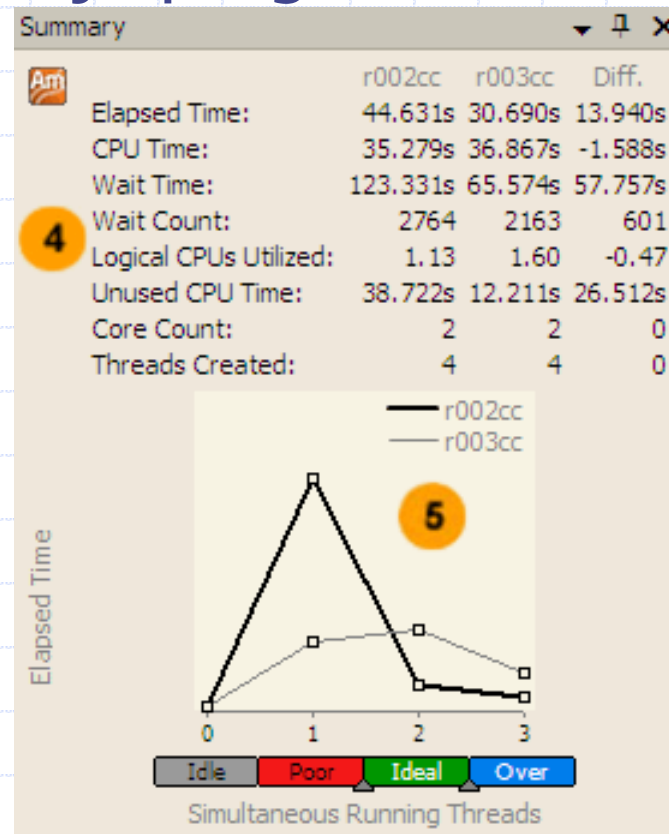
◆ 1: početno rešenje, 2: optimizovano, 3: razlika



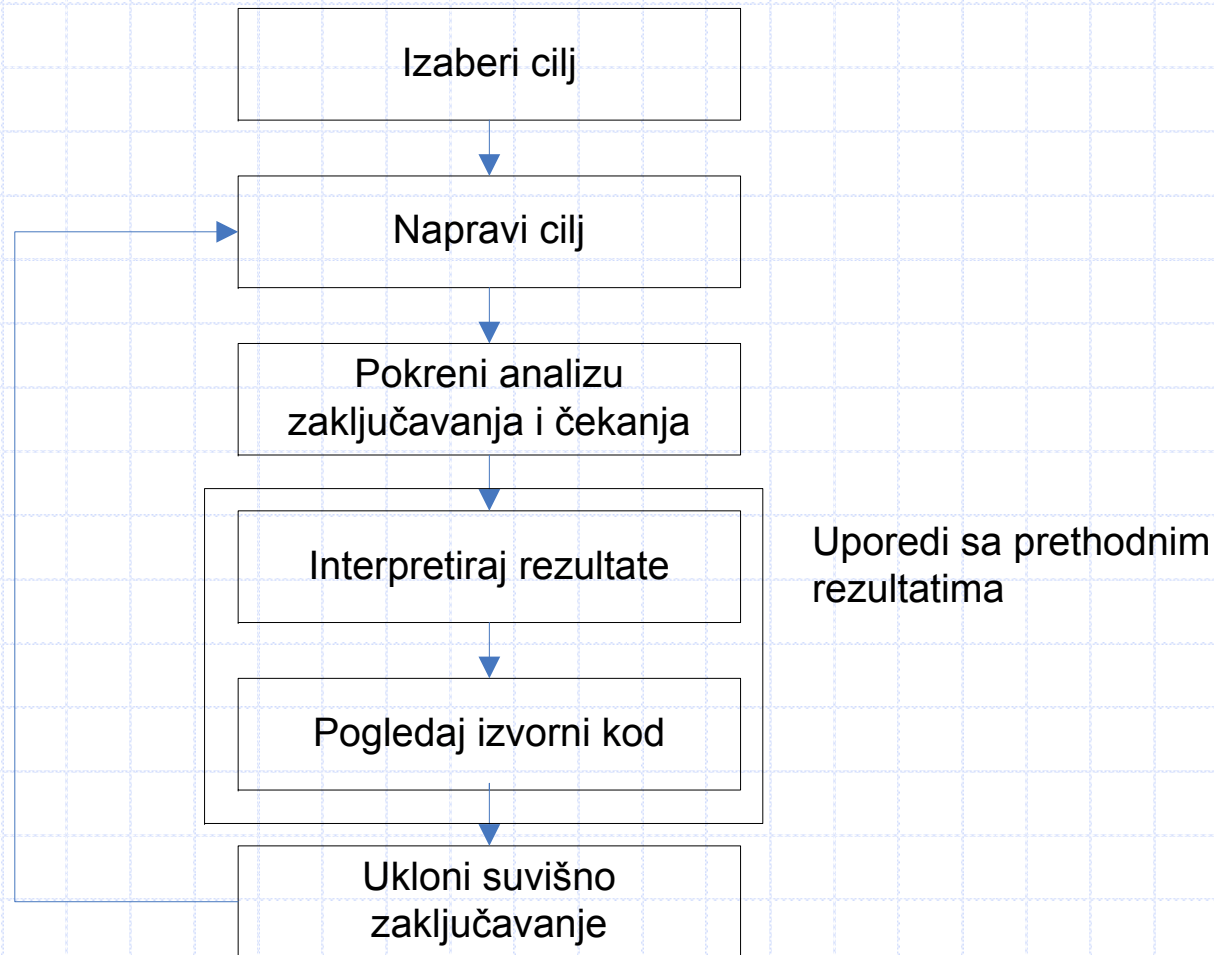
# Primer: identify\_concurrency

## 4a. Upoređenje prosečne upotrebe CPU

◆ Def. Iskorišćenih logičkih CPU = br. iskorišćenih jezgara tokom izvršenja programa



# Parallel Amplifier: Analiza zaključavanja i čekanja

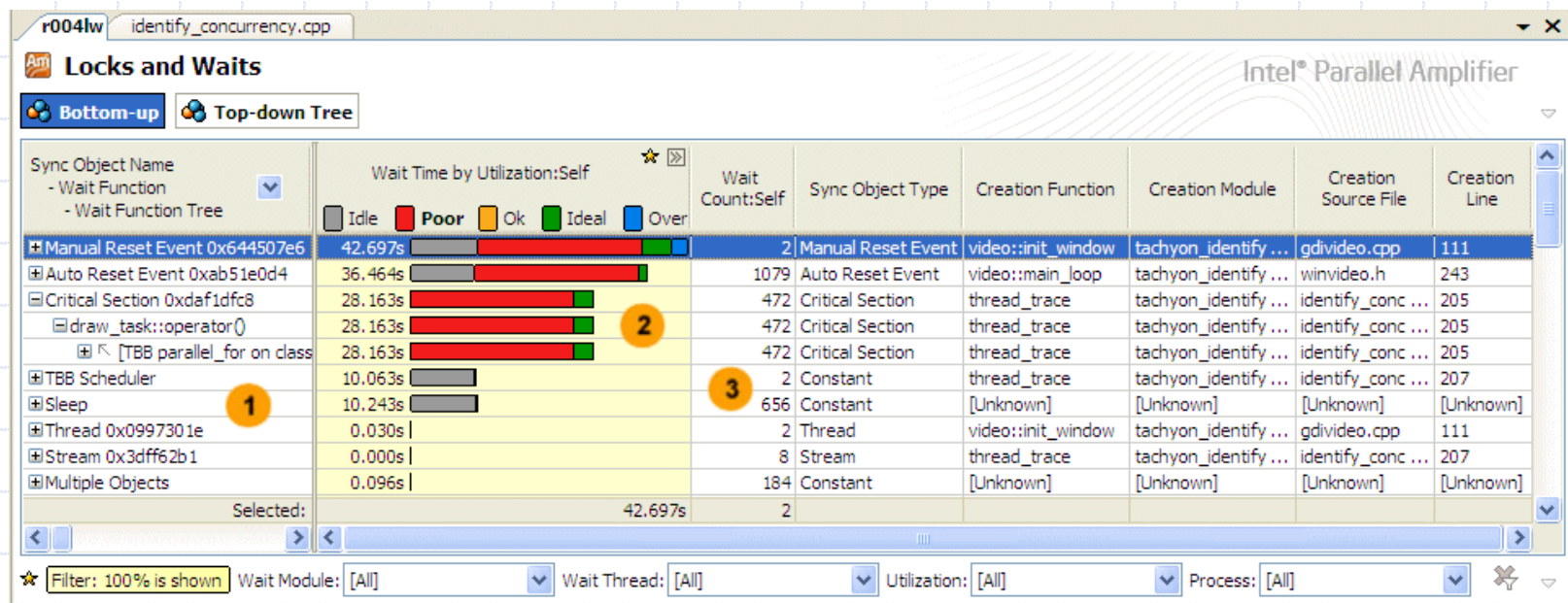


# Primer: analyze\_locks

1. Opcija: Where is my program waiting

◆ 2: vreme čekanja po iskorišćenju

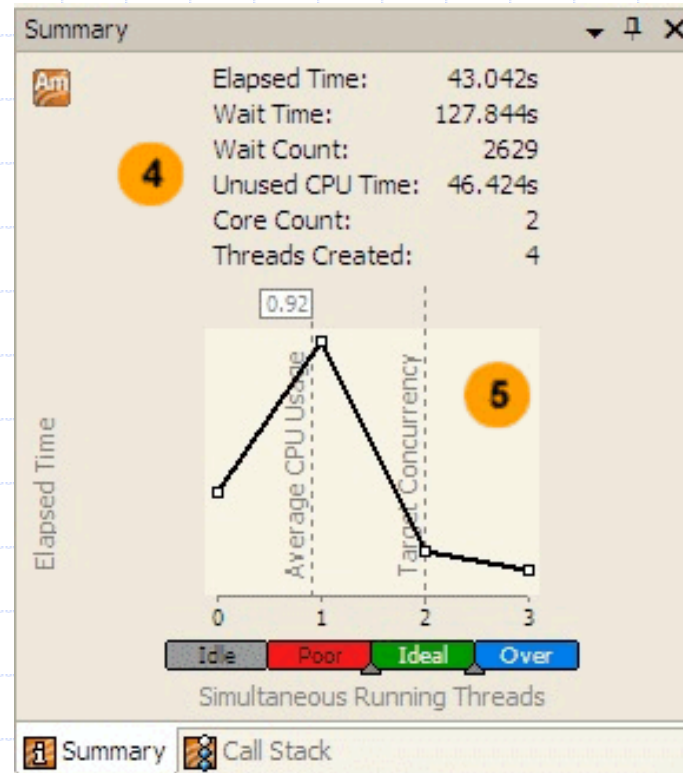
◆ 3: broj poziva API funkcije za čekanje (wait)



# Primer: analyze\_locks

## 2. Analiza sumarnog izveštaja

- ◆ 4: Wait Time je ukupno vreme čekanja niti radi sinhronizacije ili završetka U-I radnje





# Primer: analyze\_locks

## 3. Uočavanje problema i rešenje

- ◆ Analizom Bottom-up prozora uočava se da prve tri stavke nose najviše vremena čekanja
  - Prve dve su sistemske, i to na niskom nivou
  - Treća je kritična sekcija, dakle bliže aplikaciji
    - ◆ Kada se otvori, vidi se da je poziva funkcija draw\_task
- ◆ Analizom draw\_task vidi se da se čeka na zaključavanju nepotrebne kritične sekcije
- ◆ Rešenje: uklanjanje kritične sekcije
  - Dalje se upoređuju prethodno i optimizovano rešenje kao u prethodnim primerima