upcomillas es

upcomillas es

7-Reliability and performance

Advanced Computing Tools for Applied Research

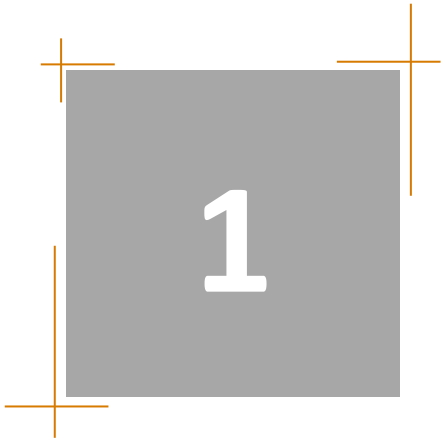
(*Herramientas Computacionales Avanzadas para la Investigación Aplicada*)

Rafael Palacios, Jaime Boal

Contents

Implementing computational tools

1. Software Reliability
2. Software Performance
3. Performance tools
4. Performance examples



Software Reliability

Definition of Software Reliability

- Reliable=
  - Worthy of trust, dependable.
  - Yielding the same or compatible results.

- Software reliability
  - "The probability of failure-free operation of a computer program in a specified environment for a specified time"

  JD. Musa, A. Iannino, and K. Okumoto, *Engineering and Managing Software with Reliability Measures, McGraw-Hill, 1987.*

Reliability recipes

- Development
  - Develop your code in modules
  - Separate algorithms from the interface
  - Develop small programs to test each module

- Typical checklist
  - Are loop indexes range-tested?
  - Is input data checked for range errors?
  - Is divide-by-zero avoided? (sqrt, log, tan…)
  - Is exception handling provided?

Clarity vs performance

- Some standard "compact" instructions **may** increase performance

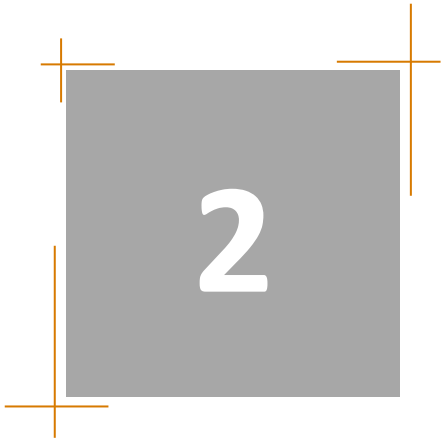
  i++; i--; → Assembly INC, DEC

  sum += x * y; → Multimedia assembly instructions:
  MLA (Multiply with accumulate) or
  MAD (Multiply and add)

- In general "compact" expressions **do not** increase performance and impact the readability of the code

Try it

sum+=*p++; vs sum += p[i];



Software Performance

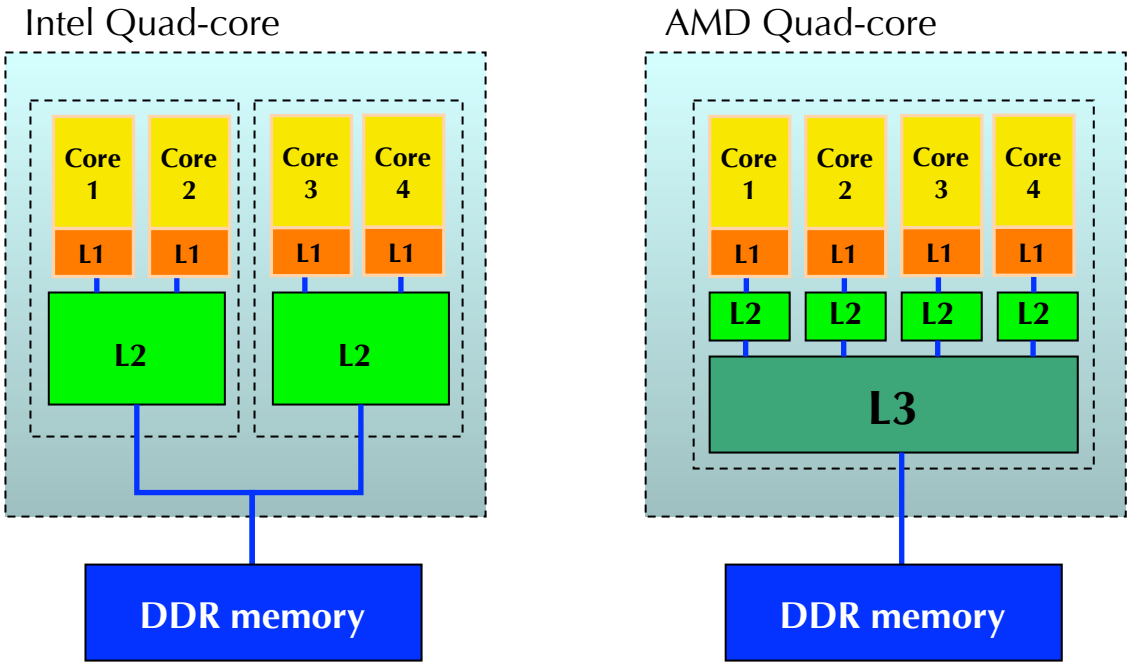
Computing performance

- It's all about locality

- Levels of locality:
  - Cache memory
  - RAM memory
  - External access

Memory access

- Cache is a small but very fast memory.
- It increases performance, but the final result depends mostly on the application and also on the configuration/size of the cache.
- Nowadays there is L1, L2 and L3 cache memory



Note:
The new Intel i7 Extreme also has independent L2, and L3:
L2: 4x256 kB
L3: 8 MB

0.5 ns

1 ns

5 ns

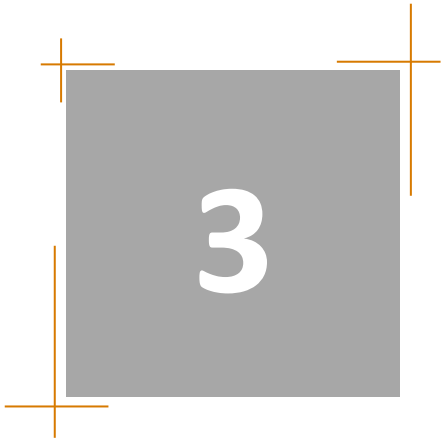
60 - 100 ns
local remote

External access

- Access to hard drive (cache 1 ns, RAM 100 ns, Disk 10,000,000 ns)
  - Data file read/write. There is another cache in the hard drive
  - Virtual memory

- Operating system calls
  - Memory allocation
  - Display information

- Server-client communications
  - Database calls
  - Web services



Performance Tools

Measuring time: C

- C functions for measuring time

```
#include <ctime>
time_t time( time_t *time ); //seconds since Jan 1st, 1970
double difftime( time_t time2, time_t time1 ); //elapsed time
```

- Example

```
time_t t0;

t0=time();
//your code goes here
t1=time();
printf("Elapsed time: %f s\n",difftime(t1,t0);
```

Measuring time: Matlab

- Matlab functions for measuring time

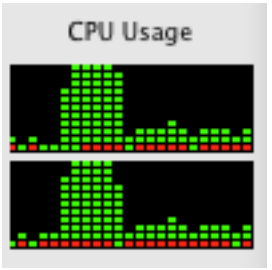
```
tic  %initialize time counter
toc  %displays elapsed time in seconds

t=cputime;  %measures CPU time. In case of parallel processing
  it sums the cpu times of all jobs.
```

- Example

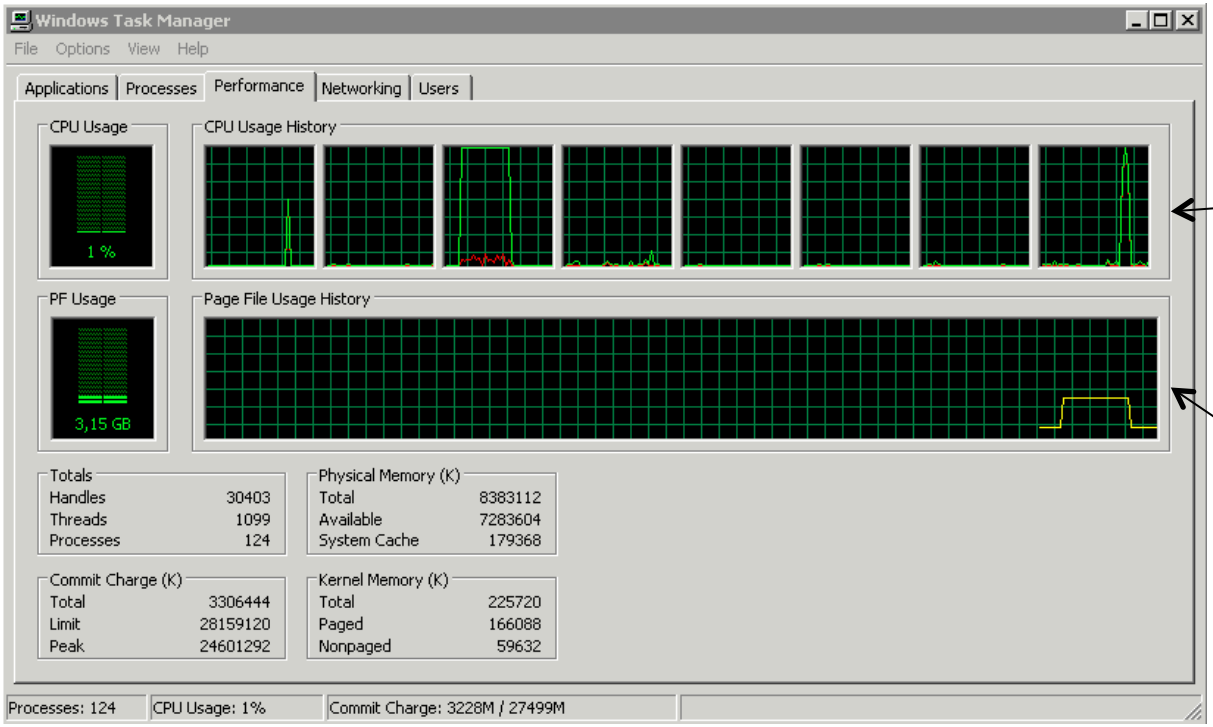
```
>> t=cputime; inv(inv(rand(3000))); cputime-t
ans =
   17.7500

>> tic, inv(inv(rand(3000))); toc
Elapsed time is 10.894806 seconds.
```



System Monitoring

- Windows Task Manager

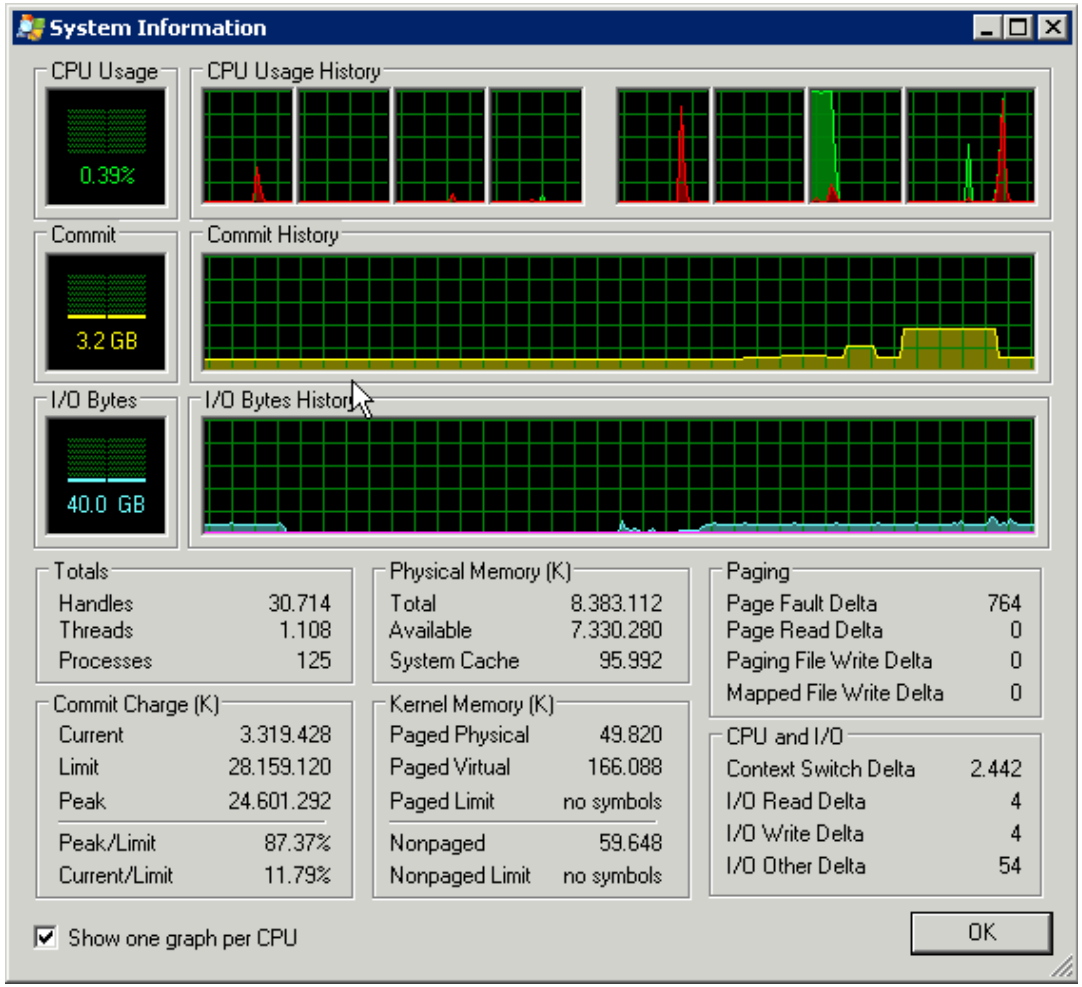


CPU usage

Memory usage

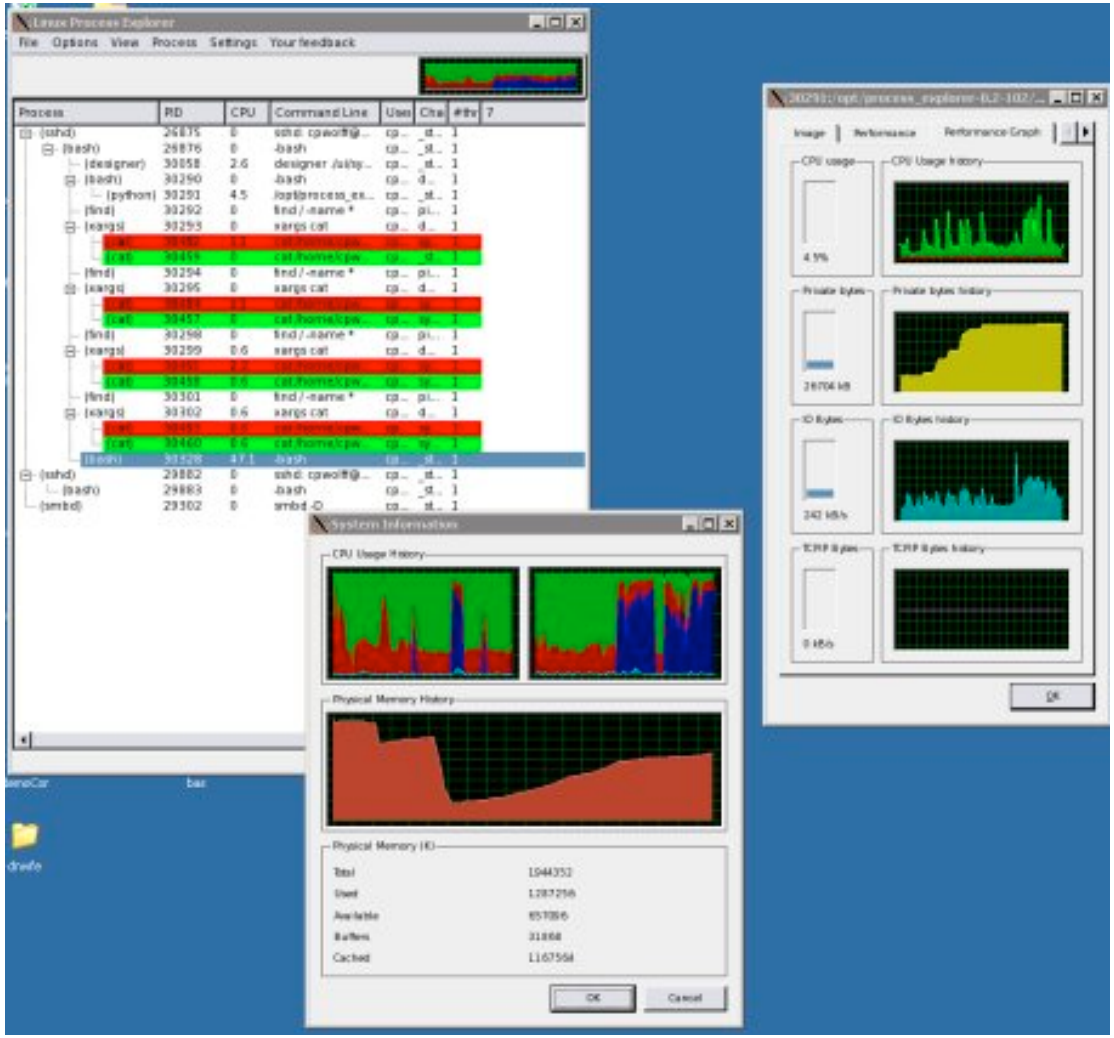
System Monitoring

- ProcExp: Developed by sysinternals.com, bought by Microsoft
  Available at: http://www.iit.upcomillas.es/ftp/pub/win/utils/



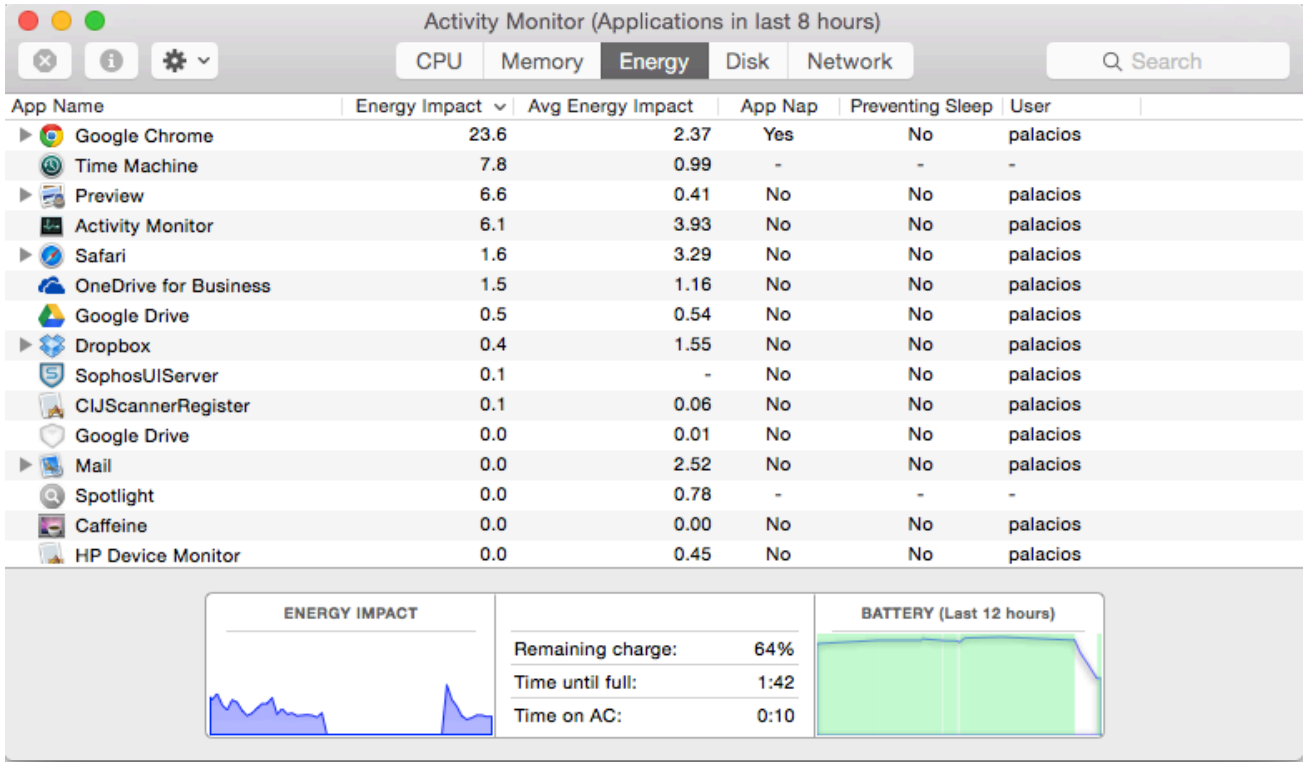
System Monitoring

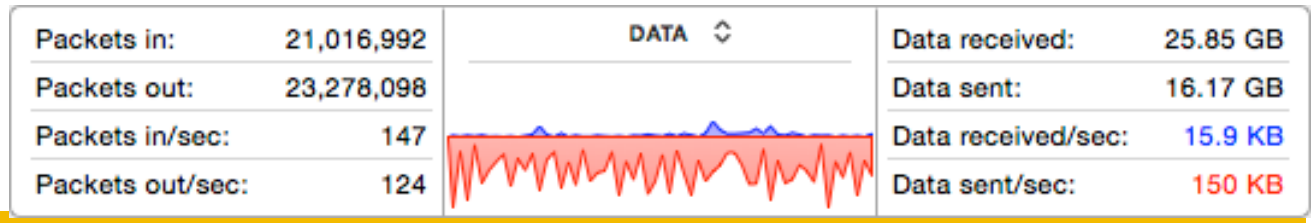
- ProcExp for linux (sourceforge.net)



System Monitoring

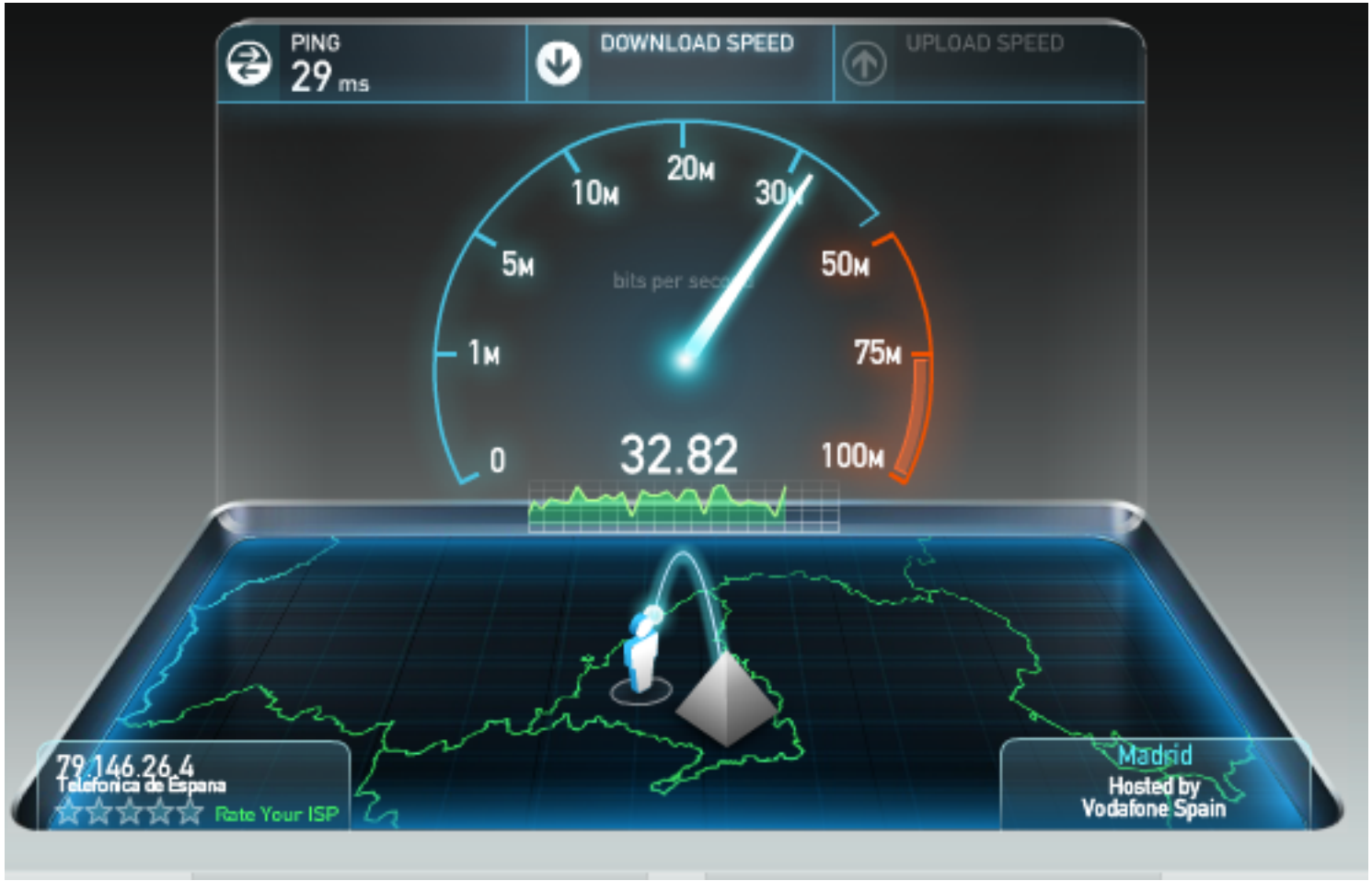
- Mac OS X Activity Monitor





Network speed

- SpeedTest.net by Ookla



Profiler

- A profiler is a tool to measure performance.
- Profilers display execution times with different levels of detail.
- They are the most useful tool to identify which sections of the code deserve the effort.

Matlab profiler

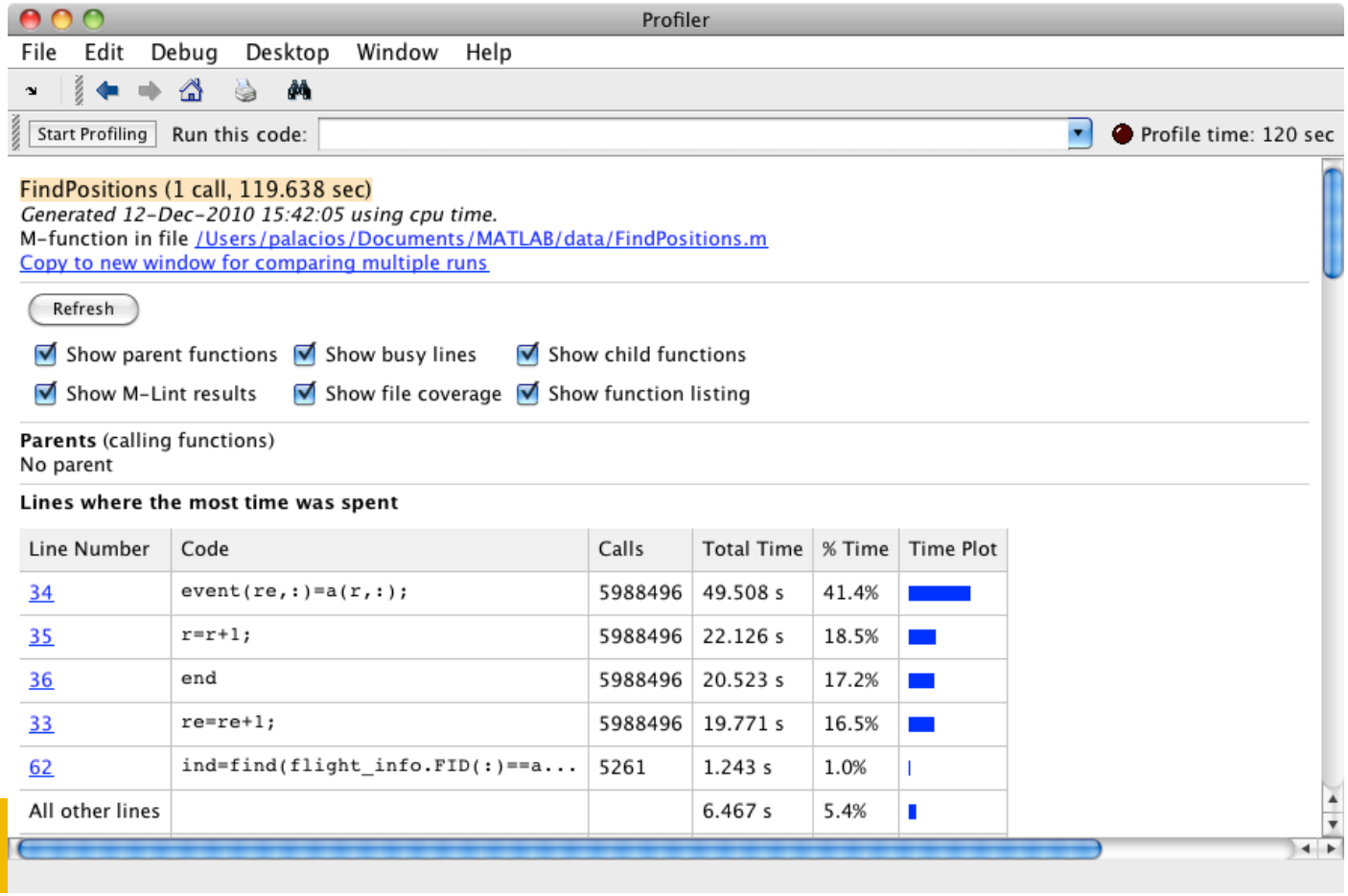
- Basic commands:

>profile on

>*your commands here*

>profile off

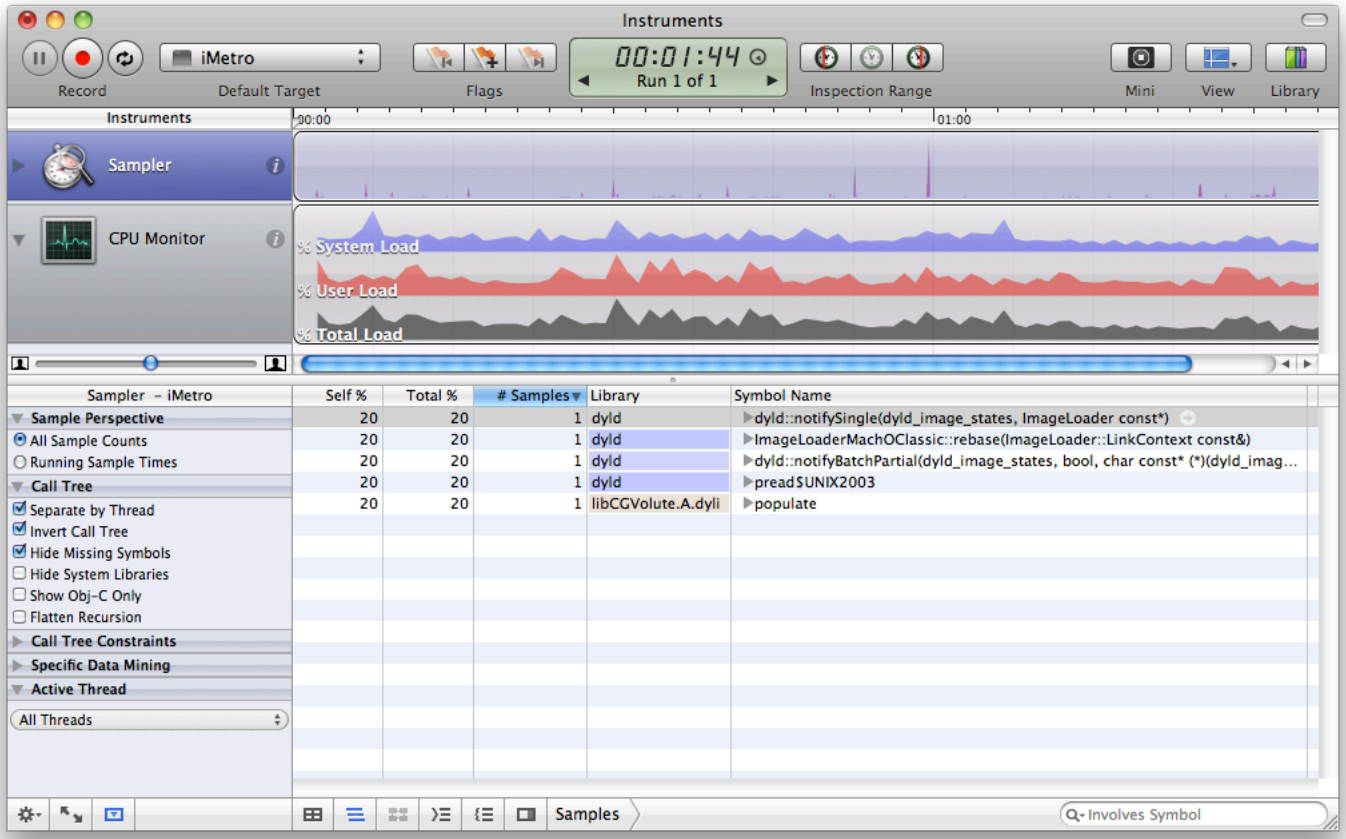
>profile report



Xcode profiler

Xcode
IPhone Software/Hardware profiler

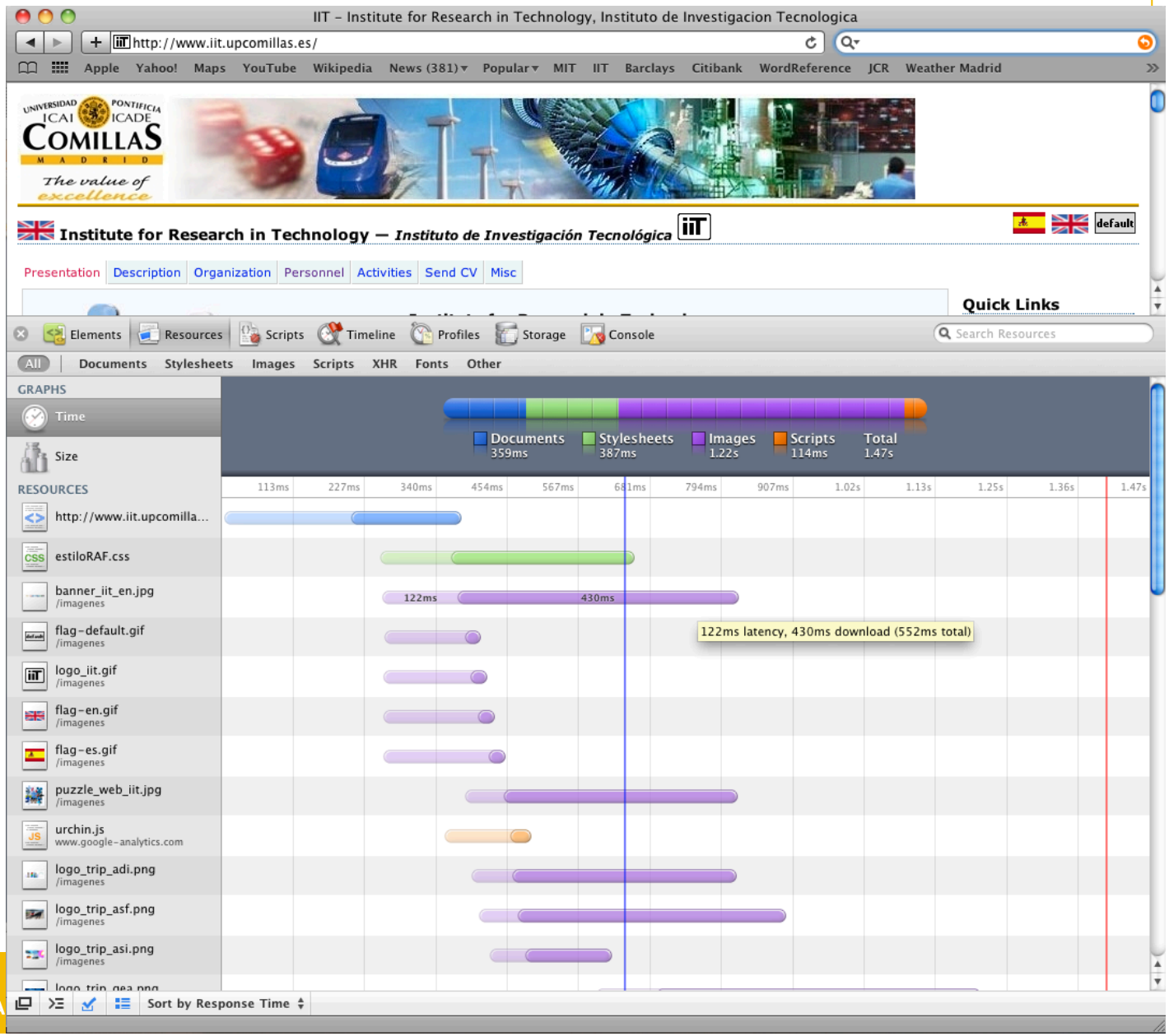
-CPU sampler
-Object allocation
-Memory leaks



Web profiler

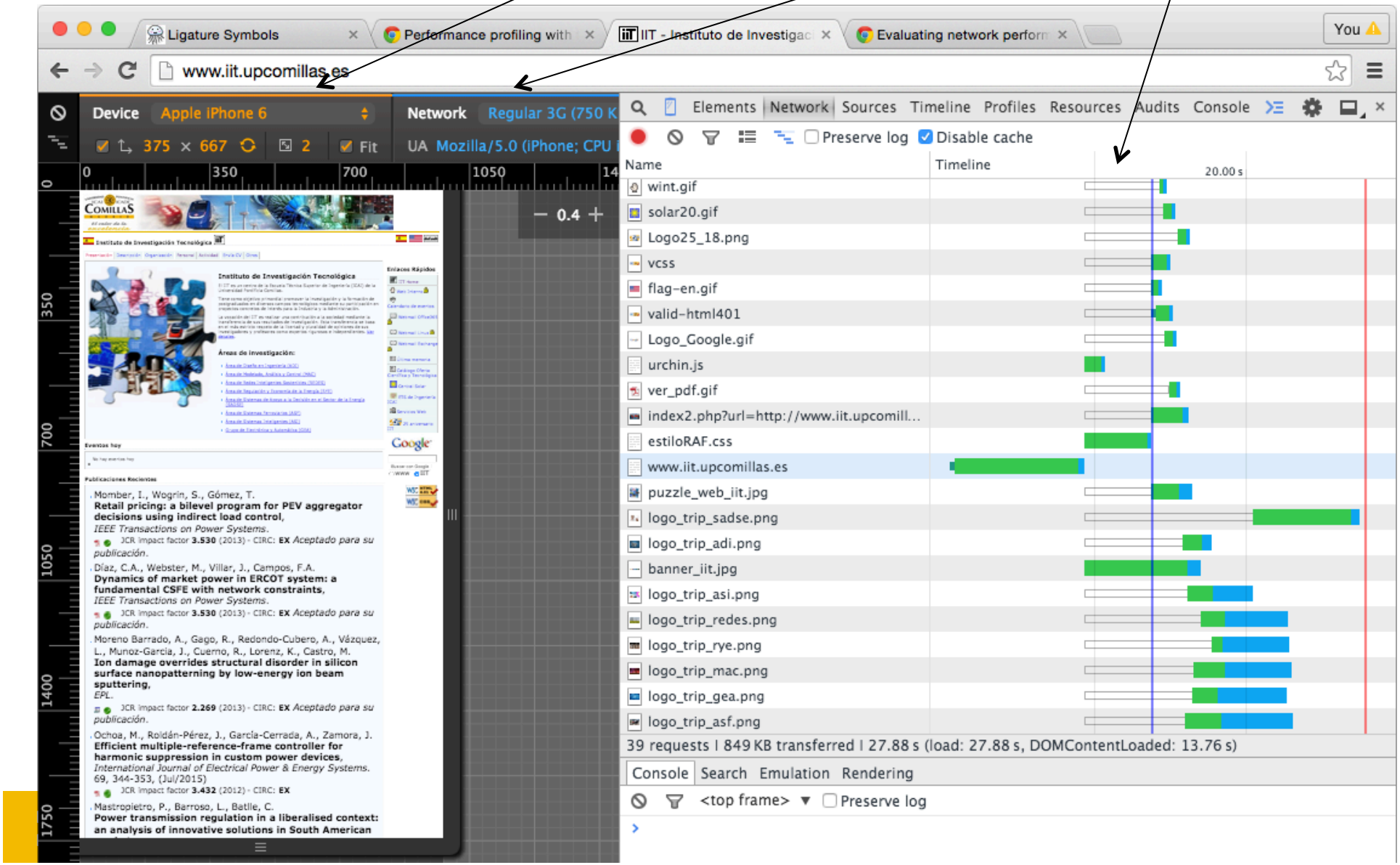
Safari
Develop>Show web inspector

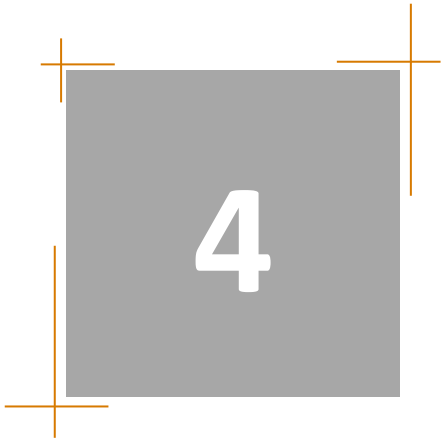
-Loading time graph
-Object sizes
-JavaScript profiler



Web profiler

Chrome. View->Developer Tools





Performance Examples

Cache use (Matlab example)

- Browse matrix by rows or columns

```
suma=0;
for i=1:1000
    for j=1:1000
        suma=suma+z(i,j);
    end
end
```

Better performance

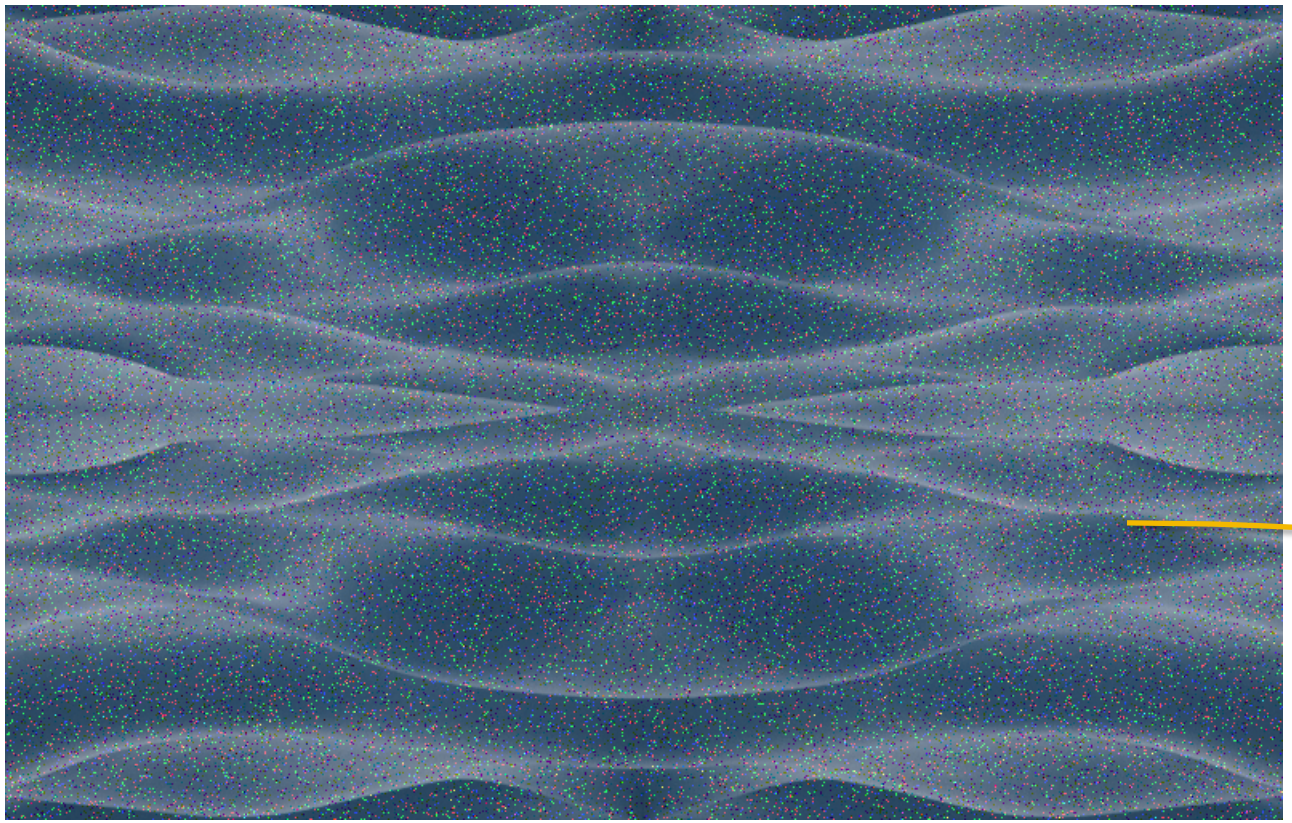
```
suma=0;
for i=1:1000
    for j=1:1000
        suma=suma+z(j,i);
    end
end
```

```
suma=sum(z(:));
```

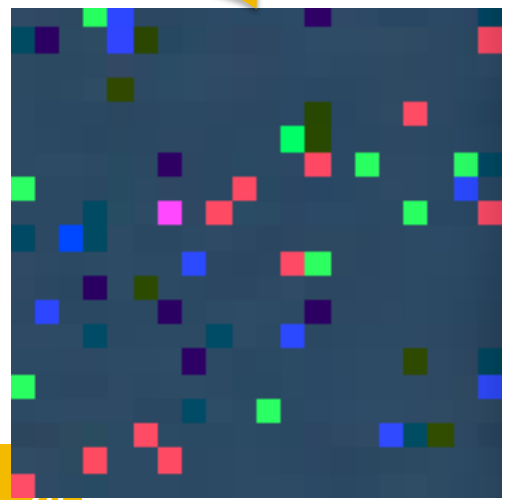
S. McGarrity, "Programming Patterns: Maximizing Code Performance by Optimizing Memory Access," *MathWorks News & Notes, Jun. 2007*

Cache use

- More sophisticated example: Median Filter



cn=imnoise(c,'salt & pepper');



Cache use

- C example of sorting algorithms

```
//Sort1: Bubble method (close exchanges)
for(x=0; x<n-1; x++)  {
   for(y=0; y<n-x-1; y++)  {
      if(array[y]>array[y+1])  {
         temp = array[y+1];
         array[y+1] = array[y];
         array[y] = temp;
      }
   }
}

//Sort2: Long exchanges
for(x=0; x<n-1; x++)  {
   for(y=x+1; y<n; y++)  {
      if(array[x]>array[y])  {
         temp = array[x];
         array[x] = array[y];
         array[y] = temp;
      }
   }
}
```

Matlab in particular

- Avoiding loops increases performance

```
>> c=imread('autumn.tif');
>> tic, for t=1:100, change, end, toc
Elapsed time is 5.6 seconds.

>> c=imread('autumn.tif');
>> tic, for t=1:100, c(find(c>200))=0; end, toc
Elapsed time is 0.06 seconds.

>> tic, for t=1:100, c(c>200)=0; end, toc
Elapsed time is 0.05 seconds.
```

100x

```
%change imagen
for i=1:size(c,1)
    for j=1:size(c,2)
        for k=1:size(c,3)
            if c(i,j,k)>200
                c(i,j,k)=0;
            end
        end
    end
end
```

- Matlab is interpreted
- Internal functions are optimized
- Internal functions may use all your cores

External calls (memory allocation)

- 1 calloc vs 1000 realloc

```
>> tic, prueba, toc
Elapsed time is 54.589000 seconds.

>> tic, for t=1:100, prueba2, end, toc
Elapsed time is 10.846000 seconds.
```

500x

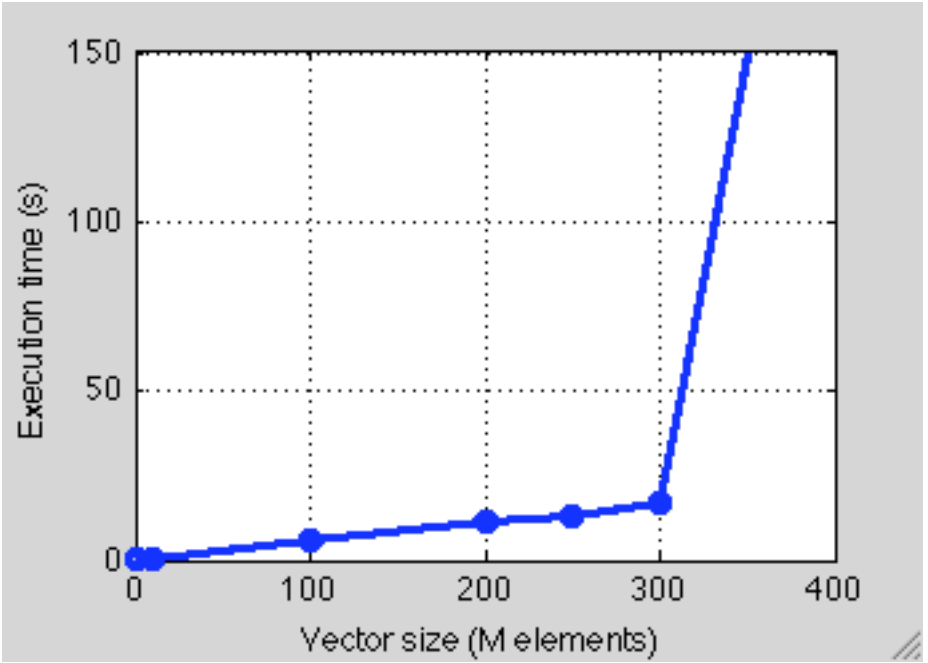
```
%prueba2
z2=ones(size(z));
for i=1:1000
    for j=1:1000
        z2(j,i)=z(j,i);
    end
end
```

```
%prueba
for i=1:1000
    for j=1:1000
        z2(j,i)=z(j,i);
    end
end
```

S. McGarrity, "Programming Patterns: Maximizing Code Performance by Optimizing Memory Access," *MathWorks News & Notes, Jun. 2007*

Virtual memory

- Just hitting the limit of the memory, execution time increases dramatically.
- Creating a new variable requires a lot of time.
- Working with a large variable is a similar problem as cache.




```
sz=350000000;
x=rand(sz,1); tic, sort(x); toc
```

Instituto de Investigación Tecnológica
C/ Santa Cruz de Marcenado, nº 26
28015 Madrid
Tel +34 91 542 28 00
Fax + 34 91 542 31 76
info@iit.upcomillas.es

www.iit.upcomillas.es

UNIVERSIDAD PONTIFICIA
ICAI ICADE
COMILLAS
MADRID