Considerations for improving program performance

Martin Gregory

PhUSE 2011
Brighton, 10-12 October 2011
Topics

- Motivation
- Diagnosis of performance problems
- Treatment alternatives
- Generally applicable treatment
- Know your programming language
- SAS-related treatment
- Conclusions
Why improve performance?
Diagnosis: possible causes

- insufficient CPU resources
- slow input/output:
  - to memory
  - to local disk
  - to network disk
- insufficient storage
  - memory
  - disk
Diagnosis: how to

- Observe the running program using appropriate tools
  - UNIX
    - ps command,
    - sosview
    - time command
  - Microsoft Windows
    - task manager
  - SAS
    - FULLSTIMER option
    - writing timing information to the log
- Examine the program carefully
  - does it do complicated calculations?
  - does it read or write a large amount of data?
  - does it break any of the rules discussed in this paper?
Diagnosis: caveats

- I/O is often hidden in “user cpu” time
- Some operating systems provide much more information
  - SAS FULLSTIMER option on both UNIX and Microsoft Windows shows:
    - real time
    - user cpu time
    - system cpu time
    - memory use
- On UNIX additional information is provided
  - page Faults
  - page Reclaims
  - page Swaps
  - voluntary Context Switches
  - involuntary Context Switches
  - block Input Operations
  - block Output Operations
Diagnosis: caveats

- Single measurements are only indicative:
  - other activity on the system might have an effect
  - especially for I/O, caching done at the system level can mask differences or even lead to false conclusions
- so:
  - Use repeated measurements
  - Measure at different times of the day and on different days
  - Use statistical analysis to check differences:
    - distributions are not normal, even after log transformations
  - No two systems are alike: your measurements are valid for the system on which you make them
Available treatment

- Add resources
  - faster cpu
  - faster i/o
  - more storage

- Tune the program
  - developer time

Both have a cost
Prevention is better than cure

• Make efficient programming techniques part of your toolbox

• Generally very low cost, but...

“We should forget about small efficiencies, say about 97% of the time: premature optimization is the root of all evil. Yet we should not pass up our opportunities in that critical 3%”.

Donald Knuth in Computing Surveys, Vol 6, No 4, December 1974
Treatment: General Rules: CPU

- Avoid unnecessary computations

```
do i=1 to &iter ;
  x=mean(1,2,3) ;
end ;
```

```
x=mean(1,2,3) ;
do i=1 to &iter ;
  end ;
```

![Bar chart showing the ratio of elapsed time inside loop to outside loop (1000 obs)]
Treatment: General Rules: CPU

- Use less expensive operations when possible

IN_0n has 190,000 obs, cvar1, cvar2 $200

cvar2 has 43,000 obs starting with 'strategically'

data out_01;
  set in_01;
  where cvar2=':strategically';
run;

data out_02;
  set in_02;
  where cvar2 contains 'strategically';
run;

data out_03;
  set in_03;
  where index(cvar2,'strategically');
run;
Treatment: General Rules: Input/Output

• Avoid unnecessary I/O –
  • be especially careful with loops
  • sorting is expensive
  • don't read or write data you don't need
Treatment: General Rules: Input/Output

Be aware of the cost of I/O operations:

- Memory
- Local disk
- Channel attached disk
- Local network disk
- WAN disk
Know your programming language: SAS vs R

%macro doit ;
  %do i=1 %to 1000 ;
    proc phreg data=sim ;
      model tanalysis*censanalysis(0)=trt ;
      output out=coxph ;
    run ;
  %end ;
%mend ;
%doit ;

for ( i in 1:1000 ) {
  coxph <- coxph(Surv(tanalysis,censanalysis==0)-trt,
                  data=sim)
Know your programming language: SAS over time

if cvar="one" then nvar=1 ;
else if cvar="two" then nvar=2 ;
else if cvar="three" then nvar=3 ;
else if cvar="four" then nvar=4 ;

if cvar="four" then nvar=4 ;
else if cvar="three" then nvar=3 ;
else if cvar="two" then nvar=2 ;
else if cvar="one" then nvar=1 ;
SAS predisposition: I/O intensive

DATA ...

PROC ...

DATA ...

SAS Dataset

External File

Log Lst

SAS Dataset

External File

Log Lst

SAS Dataset

External File

Log Lst

SAS Dataset

External File

Log Lst

Read

Write
SAS: reducing I/O: KEEP

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>275,000 Obs</th>
<th>data keep_two ; set in (keep=curr interpol) ; run ;</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURR</td>
<td>Char</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>Num</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D1RATE</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2RATE</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3RATE</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERPOL</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OBSOLETE</td>
<td>Char</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>data keep_three ; set in (keep=curr date d1rate) ; run ;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>data keep_all ; set in ; run ;</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The chart shows the comparison of execution time for different keep options, with keep=curr interpol having the shortest execution time.
### SAS: reducing I/O: WHERE vs IF

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>275,000 Obs</th>
<th>2,587 match</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURR</td>
<td>Char</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>Num</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D1RATE</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2RATE</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3RATE</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERPOL</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OBSOLETE</td>
<td>Char</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```sas
data out_if ;
  set in_if ;
  if curr='EUR';
run ;
```

```sas
data out_wh ;
  set in_wh ;
  where curr='EUR';
run ;
```

![Graph comparing execution times between 'WHERE' and 'IF']

- Where: 0.03 seconds
- Subsetting if: 0.53 seconds

The graph illustrates the performance difference between using 'WHERE' and 'IF' statements in SAS for data subset operations.
SAS: reducing I/O: DATASETS for attributes

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>275,000 Obs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURR</td>
<td>Char</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>Num</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>D1RATE</td>
<td>Num</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>D2RATE</td>
<td>Num</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>D3RATE</td>
<td>Num</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>INTERPOL</td>
<td>Num</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>OBSOLETE</td>
<td>Char</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

```plaintext
data lib.in_if;
set lib.in_if;
format d1rate 8.2;
run;
```

```plaintext
proc datasets lib=x;
modify in_wh;
format d1rate 8.2;
quit;
```

![Bar chart comparing PROC DATASETS and Data Step times](chart.png)
SAS: more efficient I/O: Indexes

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>275,000 Obs</th>
<th>Indexes:</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURR</td>
<td>Char</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DATE</td>
<td>Num</td>
<td>7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D1RATE</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D2RATE</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D3RATE</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>INTERPOL</td>
<td>Num</td>
<td>8</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OBSOLETTE</td>
<td>Char</td>
<td>1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```plaintext
data _null_; set lib.not_indexed; where ...; run;
```

```plaintext
data _null_; set lib.indexed; where ...; run;
```

![Bar chart comparing indexed and not indexed data retrieval times]
SAS: more efficient I/O: Indexes

<table>
<thead>
<tr>
<th>Variable</th>
<th>Type</th>
<th>Len</th>
<th>2,750,000 Obs</th>
</tr>
</thead>
<tbody>
<tr>
<td>CURR</td>
<td>Char</td>
<td>3</td>
<td>Indexes:</td>
</tr>
<tr>
<td>DATE</td>
<td>Num</td>
<td>7</td>
<td>curr</td>
</tr>
<tr>
<td>D1RATE</td>
<td>Num</td>
<td>8</td>
<td>curr date</td>
</tr>
<tr>
<td>D2RATE</td>
<td>Num</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>D3RATE</td>
<td>Num</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>INTERPOL</td>
<td>Num</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>OBSOLETE</td>
<td>Char</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

data _null_;  
set lib.not_indexed;  
where ... ;
run ;

data _null_;  
set lib.indexed ;  
where ... ;
run ;

---

The bar chart shows the performance difference between indexed and not indexed datasets. For 10 Observations, the indexed dataset takes approximately 1.0 seconds, while the not indexed dataset takes approximately 2.0 seconds. For 25,870 Observations, the indexed dataset takes approximately 4.5 seconds, while the not indexed dataset takes over 5 seconds.
SAS: verbosity can be expensive
SAS: trading I/O against memory

Default: BUFSIZE=0 BUFNO=1

Memory

Buffer of 80 Obs

Full dataset on disk

BUFSIZE=0 BUFNO=2

Memory

Buffer of 80 Obs

Buffer of 80 Obs

Full dataset on disk
SAS macros: nesting macro definitions

%macro outer (p1=,...) ;
   /* macro code ;
   %macro inner (ip1=,...) ;
      /* macro definition ;
      %mend ;
   %inner(ip1=x,...) ;
      /* more macro code ;
   %mend ;
%outer(p1=a,...) ;

only use if definition of %inner depends on run-time values in %outer
SAS macros: variable resolution vs function calls

```sas
%do %while (%upcase(&text)=FIXED TEXT) ;
  ...
  %end ;

%let utext=%upcase(&text) ;
%do %while (&utext=FIXED TEXT) ;
  ...
  %end ;
```
Conclusions

- there are some generally applicable rules for all circumstances
- you must know your programming language
- you must know your hardware and operating system
- measurement is critical
- 80/20 rule applies
- prevention is better than cure