Crunching Performance Data with CMS PIPELINEs

Z/VM Workshop Redux 2011
David Kreuter

Presented by:
Dave Jones

VM RESOURCES LTD
Software * Consulting * Training

VICOM Infinity

Now Showing: VM Performance - How To Turn Massive Data Into Meaningful Information

Abstract: Most of us have been challenged to produce a concise representation of the health of our VM environments. We have vast amounts of data but, for that data to be useful, we need to produce graphic charts showing resource utilization on a regular basis. The Performance Toolkit produces detailed data in reports but has a limited graphic capability. This presentation will show advanced usage of the CMS PIPELINES SPEC stage to perform summing, averaging, and other calculations on Performance Toolkit Data. This CSV data is then delivered to a workstation where it is transformed into graphs using, gulp, MSExcel. All in all a Rube Goldbergesque method nonetheless producing important data on a regular basis. Come see how SPECS, the PERFKIT Hunsberger tool, and ACUM data fit together.

Presentation Goals

- Produce charts showing meaningful performance data.
  - *MSExcel charting.*
- The performance data is in PERFKIT SUMMARY and ACUM files.
- Transform the data into Comma Separated Variable (CSV) format.
  - *Ian Hunsberger tool available from the PERFKIT web page.*
- Process performance data in CMS using PIPELINES
  - *The powerful SPECS stage*
- Works with Velocity data too!

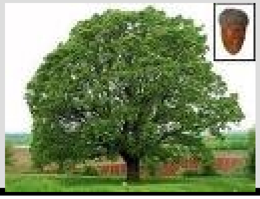
SPECS: Elsewhere in CMS?

- COPYFILE:

```
CMS COPYFILE       All Help Information                line 148 of 951
SPecs
    indicates you are going to enter a specification list to define how
    records should be copied. For more information on how you can define
    output records in a specification list, see Usage Note 10.
```

- Limited and weak as compared to the PIPELINE SPECS stage.

- *But from a single acorn a mighty oak does grow!*



COPYFILE (SPECS example

```
type cities list a
Austin
Seattle
Boston
Kansas City
Toronto
copy cities list a = newlist =  ( specs
DMSCPY601R Enter specification list:
/Cities:/ 1 1-15 9 /with SHARE conferences?/ 30

type cities newlist a
Cities: Austin               with SHARE conferences?
Cities: Seattle              with SHARE conferences?
Cities: Boston               with SHARE conferences?
Cities: Kansas City          with SHARE conferences?
Cities: Toronto              with SHARE conferences?
```

SPECS: eye ko ooh ah (ICOA)

- Basic specs: Input Conversion Output Alignment

```
type cities list a
Austin
Seattle
Boston
Kansas City
Toronto
pipe < cities list a
| specs /Cities:/ 1 1-* strip nw /with SHARE conferences?/ nw
|console
Cities: Austin with SHARE conferences?
Cities: Seattle with SHARE conferences?
Cities: Boston with SHARE conferences?
Cities: Kansas City with SHARE conferences?
Cities: Toronto with SHARE conferences?
```

***Eye ko ooh ah*?**

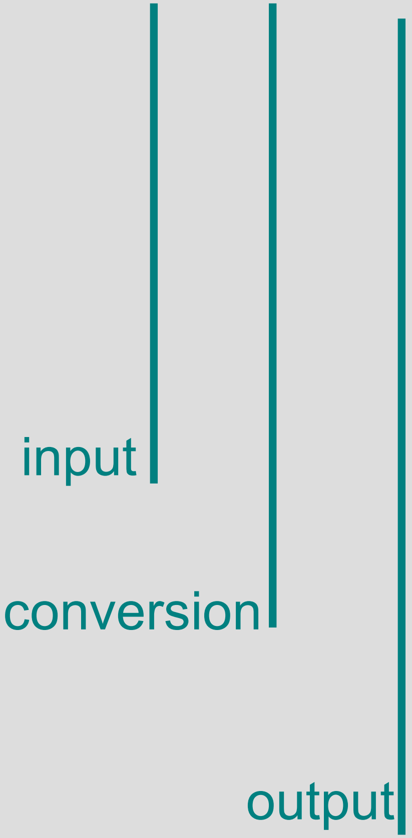
Input
Conversion
Output
Alignment

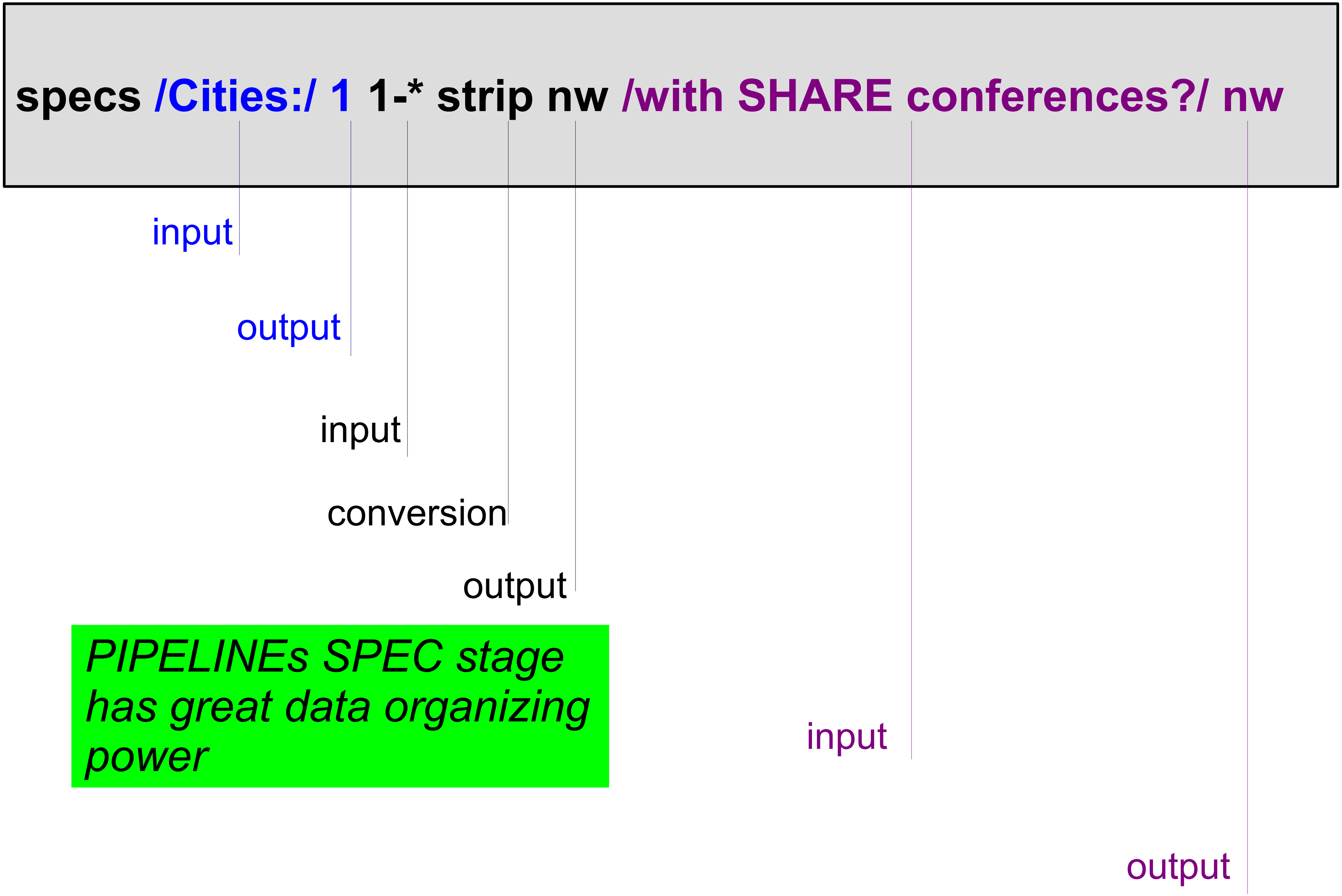
```
specs  1-*  nw.15 center 1-* c2x nw.26
```

```
 pipe < cities list a|specs  1-*  nw.15 center 1-* c2x nw.26|console
  Austin      C1A4A2A3899540404040404040
  Seattle     E28581A3A39385404040404040
  Boston      C296A2A3969540404040404040
Kansas City   D28195A281A240C389A3A84040
  Toronto     E396999695A396404040404040
```

Eye ko ooh ah?

Input
Conversion
Output
Alignment





```
specs /Cities:/ 1 1-* strip nw /with SHARE conferences?/ nw
```

input

output

input

conversion

output

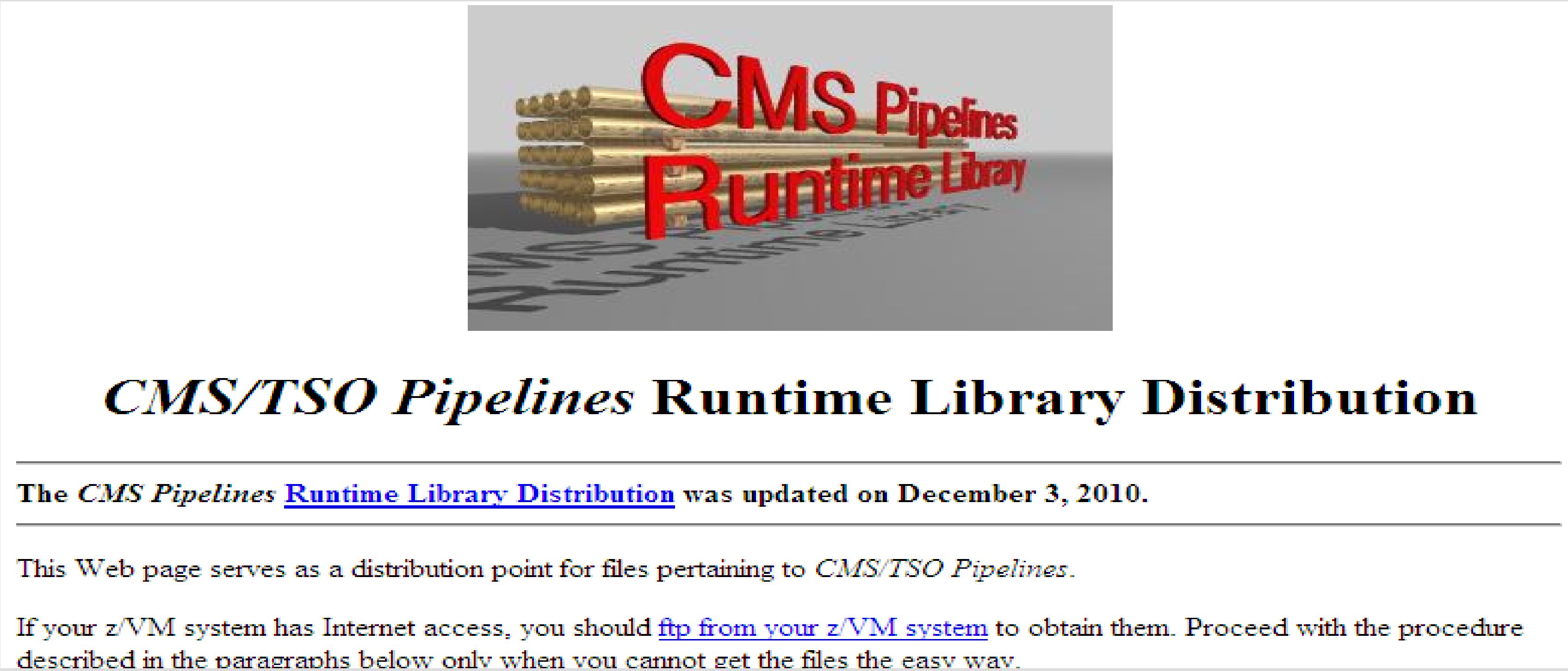
PIPELINEs SPEC stage has great data organizing power

input

output

PIPELINE Run Time Library

- Available from: http://vm.marist.edu/~pipeline/



CMS/TSO Pipelines Runtime Library Distribution

The *CMS Pipelines* Runtime Library Distribution was updated on December 3, 2010.

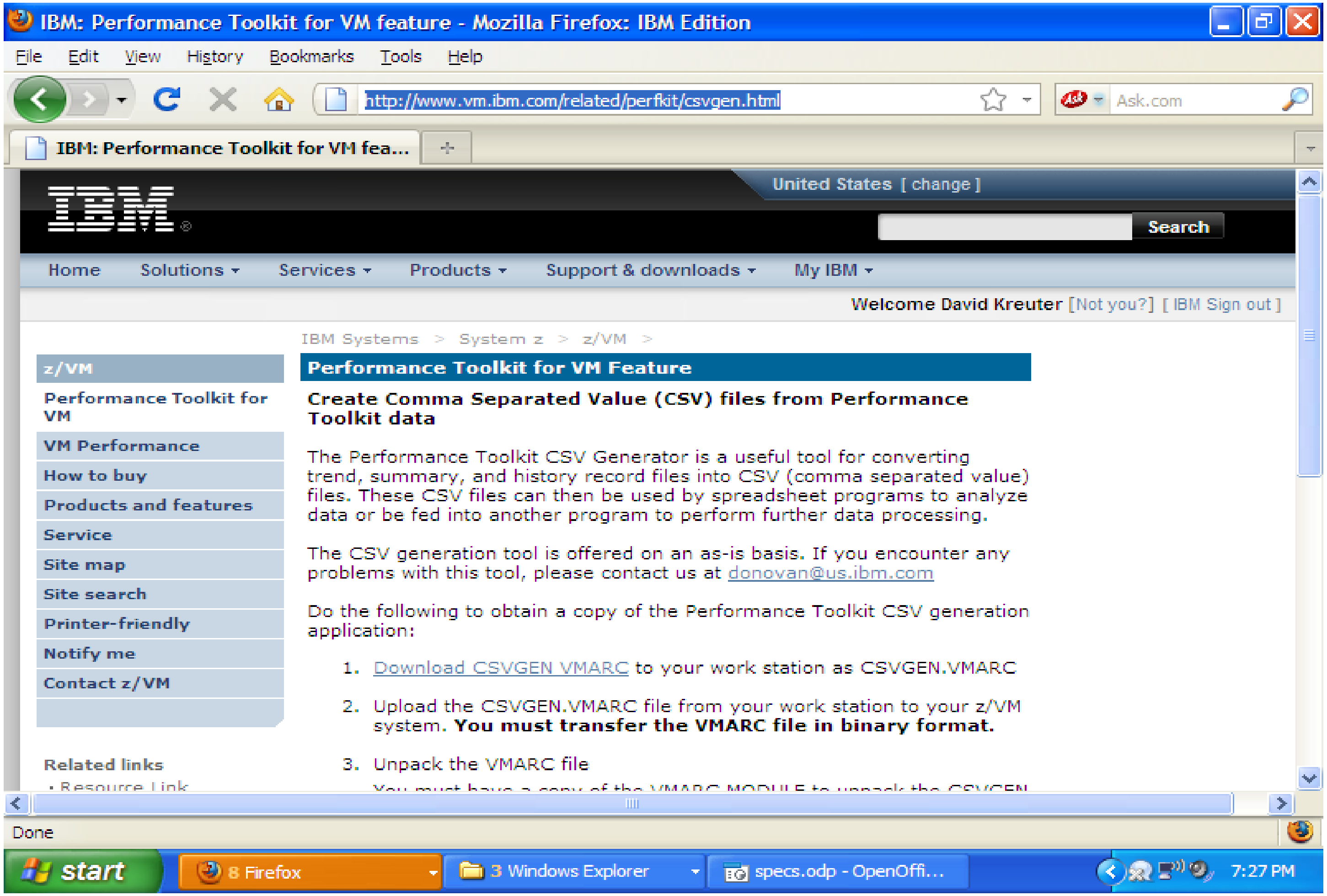
This Web page serves as a distribution point for files pertaining to *CMS/TSO Pipelines*.

If your z/VM system has Internet access, you should ftp from your z/VM system to obtain them. Proceed with the procedure described in the paragraphs below only when you cannot get the files the easy way.

Required for the niceties of SPEC

The CSVGEN Tool: required for data transformation

- Available from: www.vm.ibm.com/related/perfkit/csvgen.html


IBM: Performance Toolkit for VM feature - Mozilla Firefox: IBM Edition
File Edit View History Bookmarks Tools Help
http://www.vm.ibm.com/related/perfkit/csvgen.html
Ask.com
IBM: Performance Toolkit for VM fea...
United States [change]
Search
Home
Solutions
Services
Products
Support & downloads
My IBM
Welcome David Kreuter [Not you?] [IBM Sign out]
IBM Systems > System z > z/VM >
z/VM
Performance Toolkit for VM
VM Performance
How to buy
Products and features
Service
Site map
Site search
Printer-friendly
Notify me
Contact z/VM
Related links
• Resource Link
Done
start
8 Firefox
3 Windows Explorer
specs.odp - OpenOffi...
7:27 PM

Performance Toolkit for VM Feature

Create Comma Separated Value (CSV) files from Performance Toolkit data

The Performance Toolkit CSV Generator is a useful tool for converting trend, summary, and history record files into CSV (comma separated value) files. These CSV files can then be used by spreadsheet programs to analyze data or be fed into another program to perform further data processing.

The CSV generation tool is offered on an as-is basis. If you encounter any problems with this tool, please contact us at donovan@us.ibm.com

Do the following to obtain a copy of the Performance Toolkit CSV generation application:

1. Download CSVGEN VMARC to your work station as CSVGEN.VMARC
2. Upload the CSVGEN.VMARC file from your work station to your z/VM system. **You must transfer the VMARC file in binary format.**
3. Unpack the VMARC file
   You must have a copy of the VMARC MODULE to unpack the CSVGEN

```
pipe cms vmarc list csvgen vmarc b
|specs w1.2 1.22 read w1.2 nw.22 read w1.2 nw .22
|cons
```

CSVGEN package contents

```
HIST     COPY         SP_FCA2  COPY         SP_FCA4  COPY
SP_FCA6  COPY         SP_FCA7  COPY         SP_FCA8  COPY
SP_FCA9  COPY         SP_FC0A  COPY         SP_FC0B  COPY
SP_FC00  COPY         SP_FC01  COPY         SP_FC02  COPY
SP_FC03  COPY         SP_FC04  COPY         SP_FC05  COPY
SP_FC06  COPY         SP_FC07  COPY         SP_FC08  COPY
SP_FC09  COPY         SP_FC3A  COPY         SP_FC3C  COPY
SP_FC3E  COPY         SP_FC41  COPY         SP_FC42  COPY
SP_FC43  COPY         SP_FC44  COPY         SP_FC45  COPY
SP_FC46  COPY         SP_FC51  COPY         SP_FC55  COPY
SP_FC56  COPY         SP_FC6D  COPY         SP_FC6F  COPY
SP_FC61  COPY         SP_FC65  COPY         SP_FC68  COPY
SP_FC71  COPY         SP_STRCT COPY         SP_TCP08 COPY
SUMMARY  COPY         TRNDHEAD COPY         FINALIZE XEDIT
TOD2     EXEC         CSVGEN   EXEC         CSVGEN   PDF
```

PERFKIT Data Sources and Performance Modes

- PERFKIT processes data from the CP MONITOR DATA and from CP control blocks.
- PERFKIT does real time displays.
- PERFKIT also can save data in history and trend files.
- History and trend data can be processed by PERFKIT with the HISTDATA and TRNDSCAN commands
- *But is hard to use to produce meaningful graphic data for analysis and capacity planning purposes!*

The PERFKIT HISTSUM files

- Summary file saved on disk in ACUM HISTSUM containing one record per hour
- Controlled by:

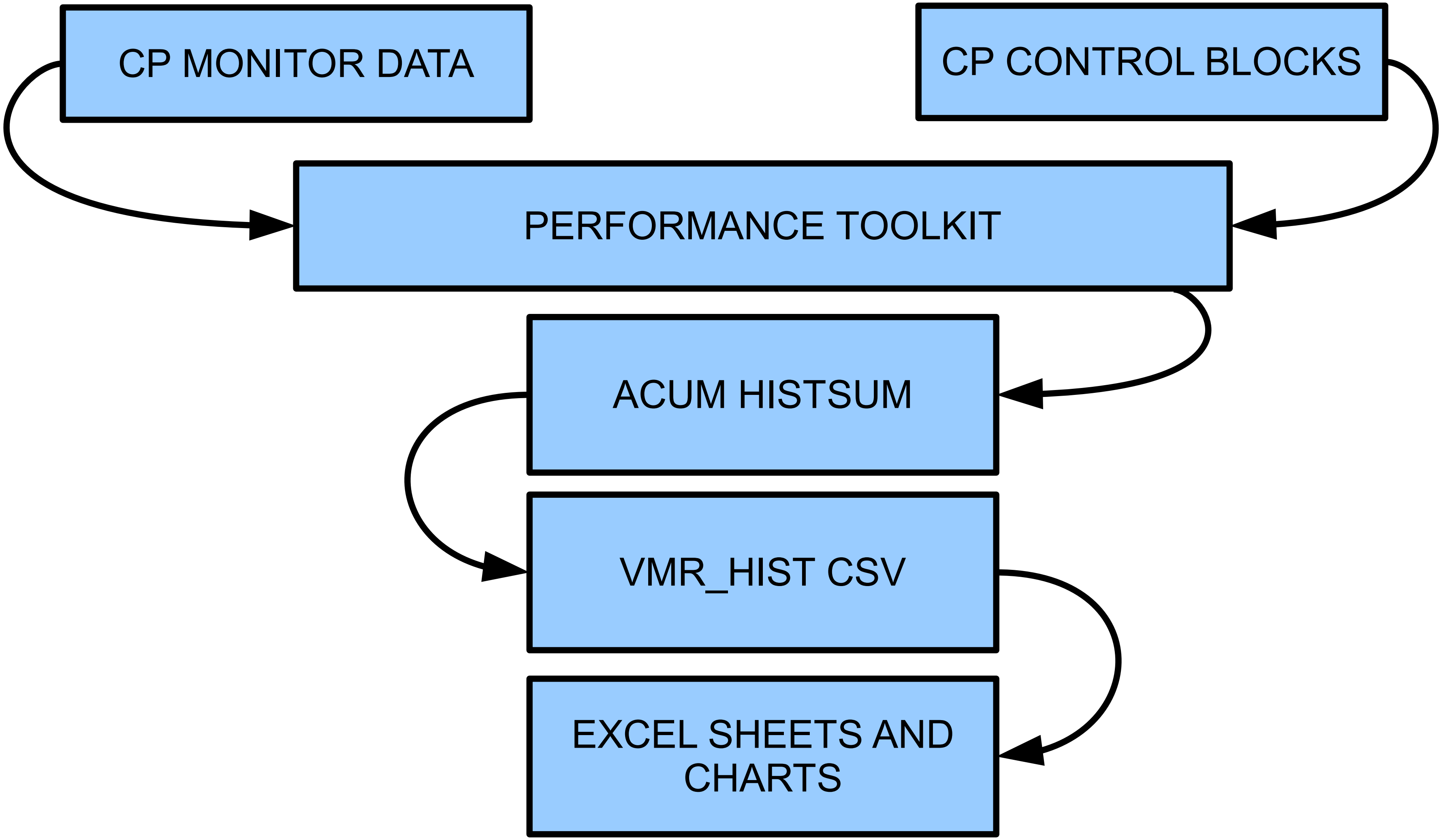
```
FCONTROL SETTINGS HISTFILE NEW
```

- Records may be tailored in the FCONX SUMREC file, default contents:

```
RECORDS   CHANNEL NSS DSPACES USER DASD SEEKS SCSI VSWITCH VNIC QDIO
RECORDS   SFS MTUSER TCPIP RSK LINUX
```

- Format of records shown in Appendix D of Performance Toolkit Reference SC24-6210-00

Data Flows



CP MONITOR DATA

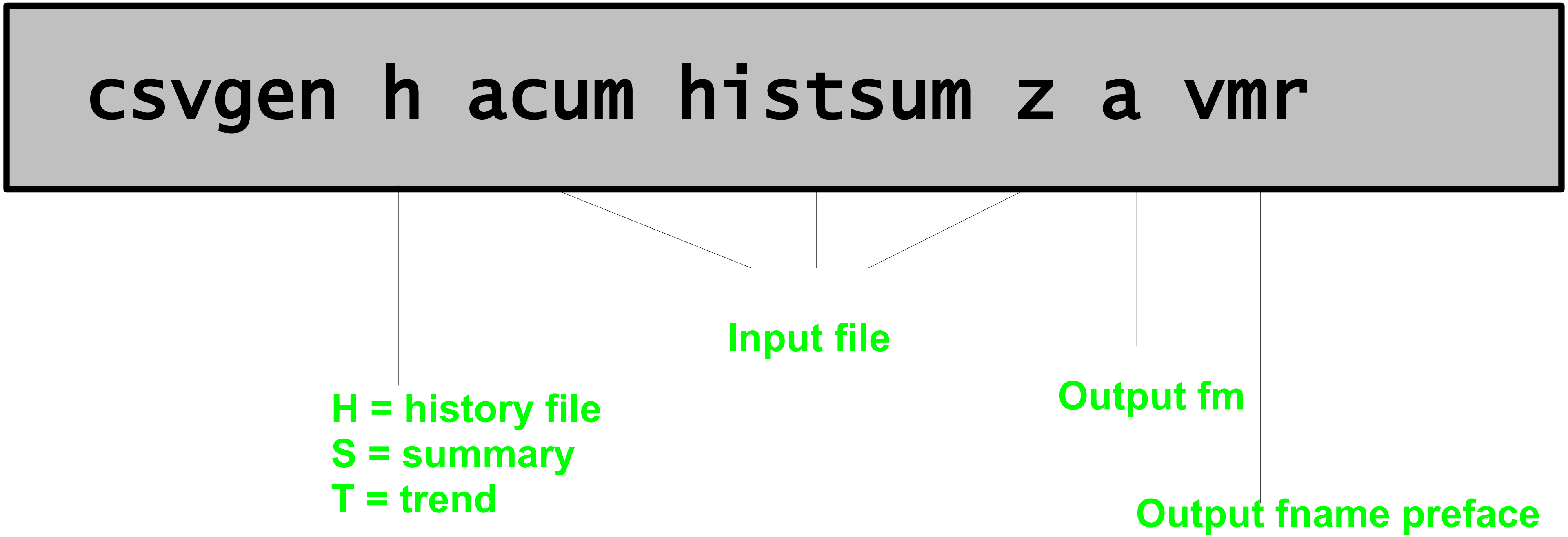
CP CONTROL BLOCKS

PERFORMANCE TOOLKIT

ACUM HISTSUM

VMR_HIST CSV

EXCEL SHEETS AND CHARTS



```
csvgen h acum histsum z a vmr
```

H = history file
S = summary
T = trend

Input file

Output fm

Output fname preface

```
CPU 00: CTIME=90:56 VTIME=005:47 TTIME=005:49
IO=081479


csvgen h acum histsum z a vmr
Ready; T=545.62/546.98 17:53:27


CPU 00: CTIME=91:07 VTIME=014:52 TTIME=014:56
IO=159383
```

CSVGEN burns a lot of CPU and does a bunch of i/o too!

CSVGEN burns a lot of CPU and does a bunch of I/O too!

```
FILENAME FILETYPE FM  FORMAT  LRECL   RECS  BLOCKS
ACUM     HISTSUM  Z1  V        1468   8513    3056

2010030410:12:03Eµõ§R y Ü   Ã{ o
2010030411:00:18Eµ¹=mßÒ Ü   Ã{ o
2010030412:00:18EµXXö ½-Ü   Ã{ o
```

Raw data

```
FILENAME FILETYPE FM FORMAT  LRECL    RECS  BLOCKS
VMR_HIST CSV      A1 F       10240    8516   21290

Date,Time,TOD,RECNO,CPUID,SYSTEMID,CPLEVEL,El_Time
Date,Time,Time-of-day,Record #,CPU serial #,VM sys
20100304,10:12:03,2010/03/04 10:12:03.848698,FC01,
20100304,11:00:18,2010/03/04 11:00:18.685342,FC01,
```

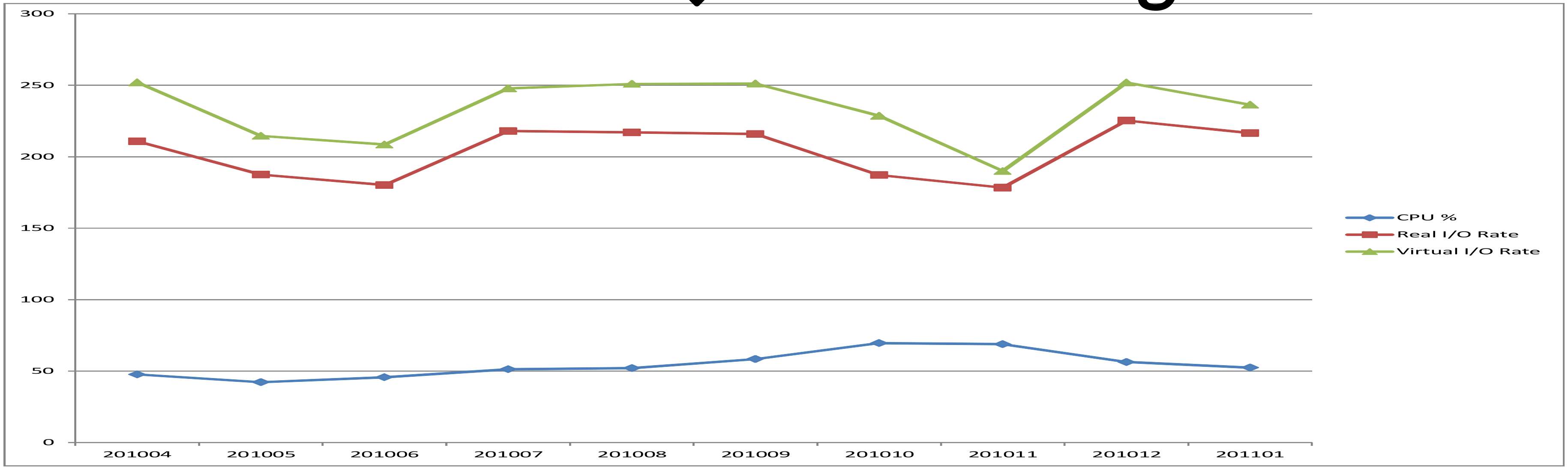
CSV data

```
FILENAME FILETYPE FM   FORMAT   LRECL    RECS   BLOCKS
ACUM     HISTSUM  Z1   V         1468    8513     3056

2010030410:12:03Eµõ§R y Ü   Ã{ o
2010030411:00:18Eµ¹=mßÒ Ü   Ã{ o
2010030412:00:18EµXXö ½-Ü   Ã{ o
```

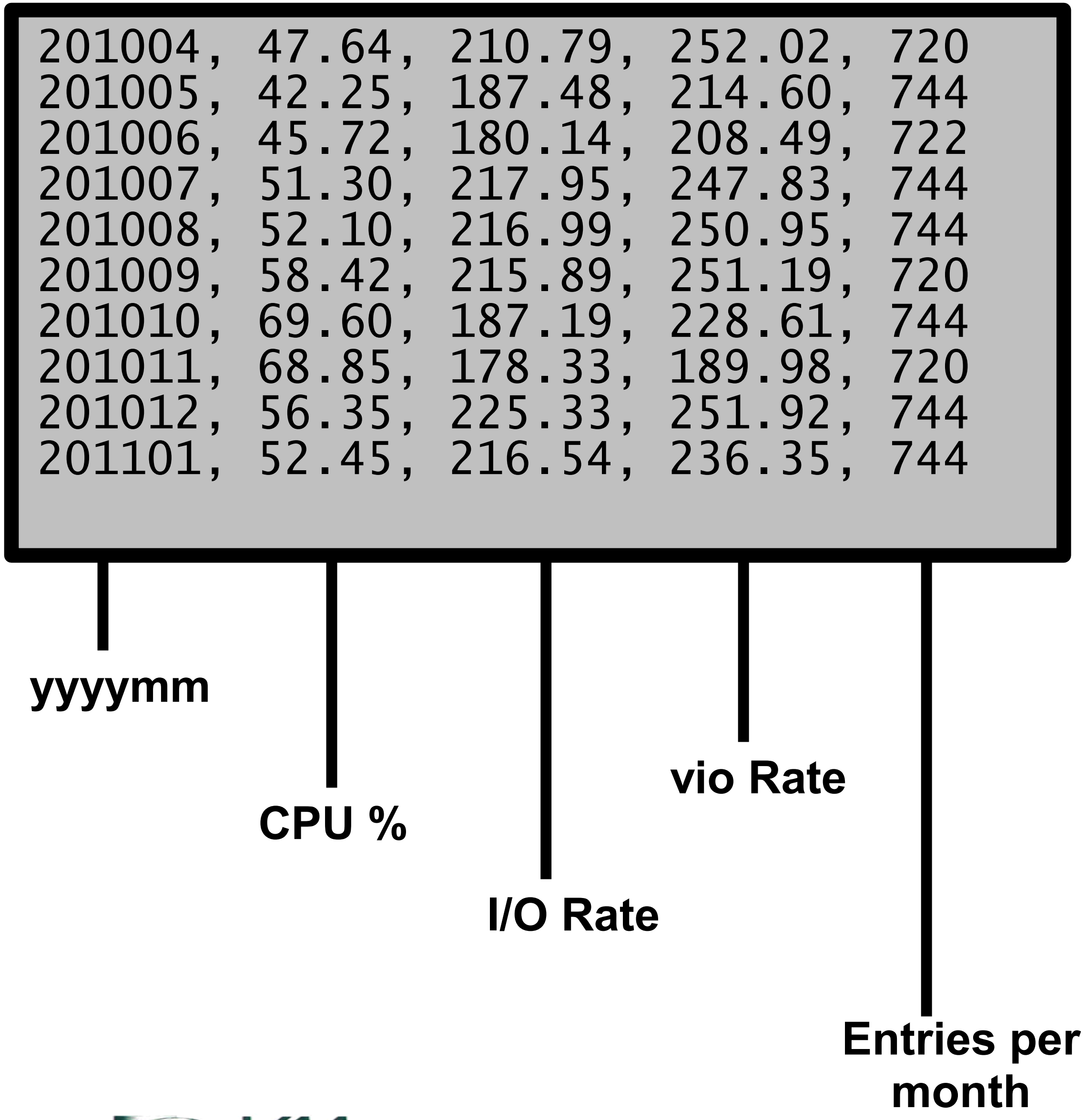
Raw data

Meaningful data



1. CMS File: TRYIO1A MONTSUM (created by the TRYIO1A EXEC)

2. Excel Spreadsheet Populated by Copy/Paste or FTP



```
201004, 47.64, 210.79, 252.02, 720
201005, 42.25, 187.48, 214.60, 744
201006, 45.72, 180.14, 208.49, 722
201007, 51.30, 217.95, 247.83, 744
201008, 52.10, 216.99, 250.95, 744
201009, 58.42, 215.89, 251.19, 720
201010, 69.60, 187.19, 228.61, 744
201011, 68.85, 178.33, 189.98, 720
201012, 56.35, 225.33, 251.92, 744
201101, 52.45, 216.54, 236.35, 744
```

yyyymm

CPU %

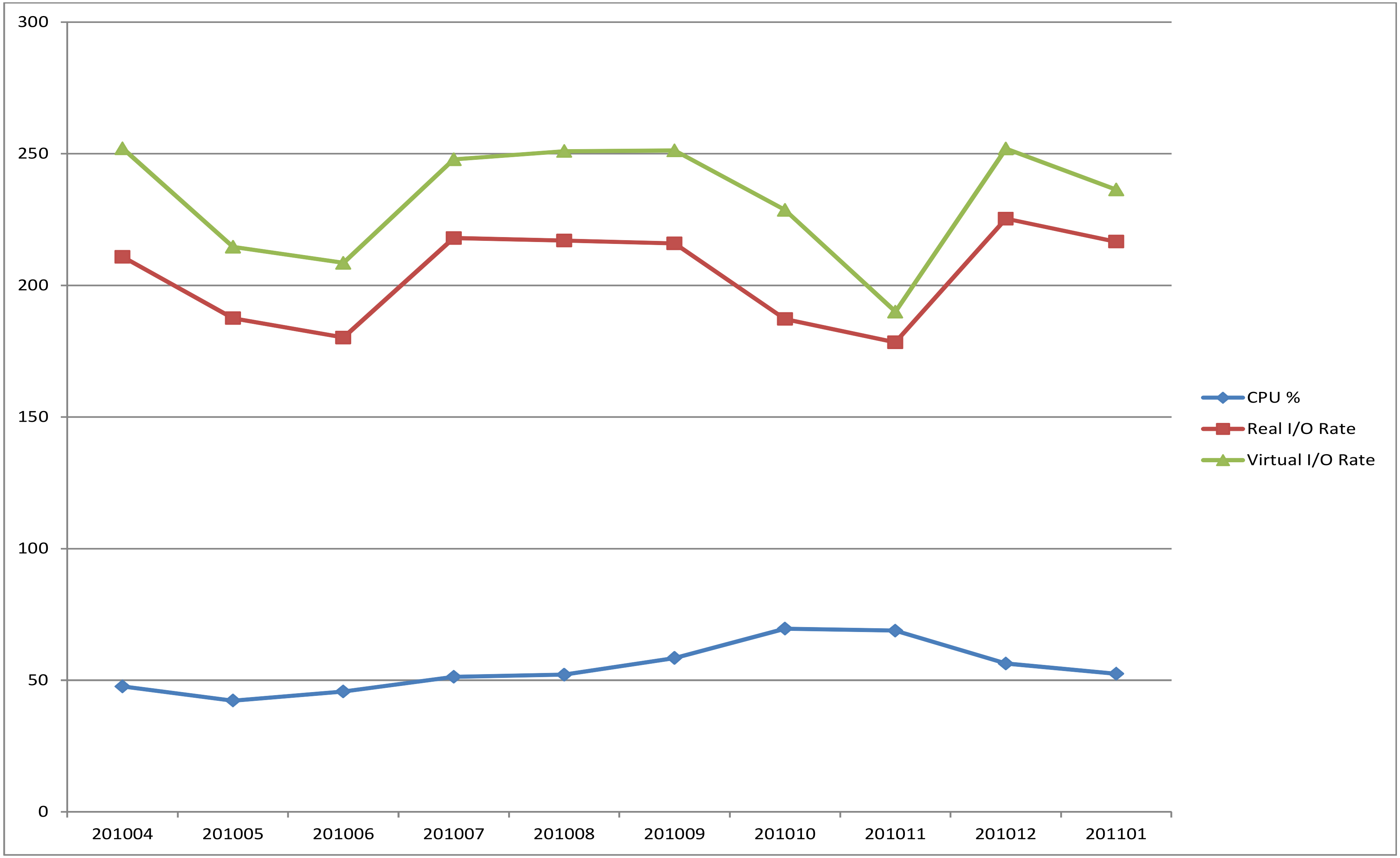
I/O Rate

vio Rate

Entries per month

yyyymm	cpu %	io rate	vio rate
201004	47.64	210.79	252.02
201005	42.25	187.48	214.6
201006	45.72	180.14	208.49
201007	51.3	217.95	247.83
201008	52.1	216.99	250.95
201009	58.42	215.89	251.19
201010	69.6	187.19	228.61
201011	68.85	178.33	189.98
201012	56.35	225.33	251.92
201101	52.45	216.54	236.35

Create a chart using EXEC charting facilities. No calculation performed in MSExcel (no formulas, macros, etc.)



The next four slides

- The code for the TRYIO1A EXEC shown.
- Do some plumbing:
  - *Read the VMR_HIST CSV*
  - *Speculate*
  - *Write out two files:*
    - *Stream 0: TRYIO1A DAILY*
    - *Stream 1: TRYIO1A MONTSUM*

Source code 1 of 4

```
/**/
parse source . . xcnm xctyp . . how .

/*
field 1: date
field 2: time
field 11: CPU percentage
field 10: # of cpus
field 25: io rate
field 46: vio rate
*/


'PIPE       (endchar ?) ',
' < VMR_HIST CSV A',
 '| DROP 2',
 '| DROP LAST',
```

Source code 2 of 4

```
' | s: specs',
   ' printonly a      ', /* print only the break record a */
   '   fieldsep , ' ',
   ' select second ',    /* use second buffer station      */
     ' a: f1  1' , /* define field a                       */
     ' b: f11 .',  /* define field b to be summed/averaged */
     ' c: 1.6 .',  /* yyyymm                               */
     ' h: f25 .',  /* io rate                              */
     ' i: f46 .',  /* vio rate                             */
     ' q: f10 .',  /* number of cpus                       */
     ' set #0+=b', /* compute CPU into counter 0           */
     ' set #1+=b', /* compute CPU into counter 1           */
     ' set #2+=1', /* how many items summed into counter 2 */
     ' set #3+=1', /* how many items summed into counter 3 */
     ' set #10+=i', /* io count */
     ' set #11+=i', /* io count */
     ' set #12+=h', /* io count */
     ' set #13+=h', /* io count */
     ' break a',     /* break on changes to a */
' break a',     /* break on changes to a */
' print ((#0/#2)/q; #0:=0) picture zzz9.99 strip nw',
     ' /,/ N ',
     ' print (#2; #2:=0) picture zzzz9 nw ',
     ' /,/ N ',
     ' print (#10; #10:=0) picture zzzz9 nw ',
     ' /,/ N ',
     ' print (#12; #12:=0) picture zzzz9 nw ',
     ' /,/ N ',
     ' write ' ,
     ' break c',
     'if #3>=(28*24)',
     ' then ',
         ' print c 1.6 left',
         ' /,/ N ',
         ' print ((#1/#3)/q; #1:=0) picture zzz9.99 strip nw',
         ' /,/ N ',
         ' print ((#11/#3)/q; #11:=0) picture zzz9.99 strip nw',
         ' /,/ N ',
         ' print ((#13/#3)/q; #13:=0) picture  zz9.99 strip nw',
         ' /,/ N ',
         ' print (#3; #3:=0) nw.3 right',
         ' outstream 1',
      'else ',
         'set (#3:=0;#1:=0;#11:=0;#13:=0)',
      'endif',
```

Source code 3 of 4

Source code 4 of 4

```
     'if #3>=(28*24)',
     ' then ',
         ' print c 1.6 left',
         ' /,/ N ',
         ' print ((#1/#3)/q; #1:=0)  picture zzz9.99 strip nw',
         ' /,/ N ',
         ' print ((#11/#3)/q; #11:=0) picture zzz9.99 strip nw',
         ' /,/ N ',
         ' print ((#13/#3)/q; #13:=0) picture  zz9.99 strip nw',
         ' /,/ N ',
         ' print (#3; #3:=0) nw.3 right',
         ' outstream 1',
      'else ',
         'set (#3:=0;#1:=0;#11:=0;#13:=0)',
      'endif',
' | > ' xcnm 'DAILY  A',
'?',
's:',
' | > ' xcnm 'MONTSUM A'
```

SPECing concepts used:

- Field separator
- Multistream output
- Alignment
- Stripping
- Counters
- Read stations
- Break records
- Printing
- Logic

```
s: specs',
< other specing >
break a',
'print ((#0/#2)/q; #0:=0) picture zzz9.99 strip
  nw',
     < other print statements>
' write  ,
'break c',
< other specing >
     ' print c 1.6 left',
  '    < other print statements
 'outstream 1',
 '| > ' xcnm 'DAILY  A',
'?',
's:',
' | > ' xcnm 'MONTSUM A'
```

Declare multistream specs (s: specs), when changes to field a (break a) print some records,
Write them to primary output stream – TRYIO1A DAILY -- (write), when changes for field c (break c) print some records, direct to output stream 1 (outstream 1), second pipe (s:) write to TRYIO1A MONTSUM.

Not all spec items shown

```
' printonly a       ', /* print only the break record a */
   '    fieldsep , ' ,
   ' select second ',    /* use second buffer station         */
     ' a: f1  1' , /* define field a yyyymmhh               */
     ' b: f11 .',  /* define field b to be summed/averaged  */
     ' c: 1.6 .',  /* yyyymm                                */
     ' h: f25 .',  /* io rate                               */
     ' i: f46 .',  /* vio rate                              */
     ' q: f10 .',  /* number of cpus                        */
```

Verbatim spec coding

Print only on the break record (printonly a). The break record is a daily summary. Declare fields (a: f1 1 … q: f10 .).
Use the second buffer station (select second).
Use the comma as the field separator (fieldsep ,)

Second reading station and record breaks

```
select second
a: f1  1
< setup the record, calculations, etc >
break a
```

- After each cycle, *spec* loads the record on the primary input stream into a buffer that is called the *second reading station*, or “second reading” for short.
- Field a is the yyyymmdd.
- The control break is active while the last record having a particular key (same yyyymmdd) is being processed.
- The record that causes (not equal) the break is in the first reading station and moved to the second reading station after the break.

Second reading station and record breaks

```
select second
 c: 1.6 .     /* yyyymm */
 < other specifications >
 break c
```

- Record break in field c (yyyymm) will form output record with monthly summary records for secondary output stream (outstream 1).
- Field c is *not* in the output record.
- So a break hierarchy is created, break a for changes on yyyymmdd (daily), break c on changes on yyyymm (monthly)

Field identifiers

```
a: f1  1 , /* define field a       */
b: f11 . ,  /* define field b to be sum/avg'd*/
c: 1.6 . ,  /* yyyymm               */
h: f25 . ,  /* io rate              */
i: f46 . ,  /* vio rate             */
q: f10 . ,  /* number of cpus       */
```

- Fields are identified by a lower or upper case letter followed by a colon. There are fifty-two possible fields available to the speculative plumber.

Verbatim spec coding

Counter expressions: Calculations and reset

```
set #0+=b /* compute CPU into counter 0           */
set #1+=b /* compute CPU into counter 1           */
set #2+=1 /* how many items summed into counter 2 */
set #3+=1 /* how many items summed into counter 3 */
set #10+=i /* vio count */
set #11+=i /* vio count */
set #12+=h /* io count */
set #13+=h /* io count */
```

Almost Verbatim spec coding

- **Counter is identified as zero or positive with no limit on the number of counters. A counter commences with the # sign.**
- **Specs has an *alu* (arithmetic logic unit). The alu has many operations – showing adding field values to a counter (accumulators) using the set specification.**

Logic

```
break c
if #3>=(28*24)
      then
          print c 1.6 left /,/ N
         < more print statements >
          print (#3; #3:=0) nw.3 right
          outstream 1
      else
         set (#3:=0;#1:=0;#11:=0;#13:=0)
      Endif
```

Pruned the spec coding

- Specs has a wide range of logic and conditional capabilities. This example shows an if/then/else/endif construct testing if there are 28 or more daily records at break c. **If** there are equal to or greater than 28 days of records **then** print to outstream 1 and reset counters, **else** it is a short month (from the input) in which case reset the counters to 0

Print and pictures

```
break a     /* break on changes to a */
print ((#0/#2)/q; #0:=0) picture zzz9.99 strip nw  /,/ N
print (#2; #2:=0) picture zzzz9 nw    /,/ N
print (#10; #10:=0) picture zzzz9 nw  /,/ N
print (#12; #12:=0) picture zzzz9 nw  /,/ N
Write
```

- On the break record (a) Print to the output record by using the alu counter 0 divided by counter 2 (CPU percentage divided by number of processors), reset counter to 0, print counters 2, 10, and 12 in the next words and reset counters 2, 10 and 12 to zeroes. The picture specification controls the way a counter is formatted. The z is used to select significant digits, the 9 is used to select a digit in that position. Write to the selected output stream, default is stream 0.
- *The contents of the print records in this slide are formatted to include the /,/ n on each line.*

Print and pictures: on break c (yyyymm)

```
print c 1.6 left /,/ N
print ((#1/#3)/q; #1:=0) picture zzz9.99 strip nw, /,/ N
print ((#11/#3)/q; #11:=0) picture zzz9.99 strip nw, /,/ N
print ((#13/#3)/q; #13:=0) picture  zz9.99 strip nw, /,/ N
print (#3; #3:=0) nw.3 right
Outstream 1
```

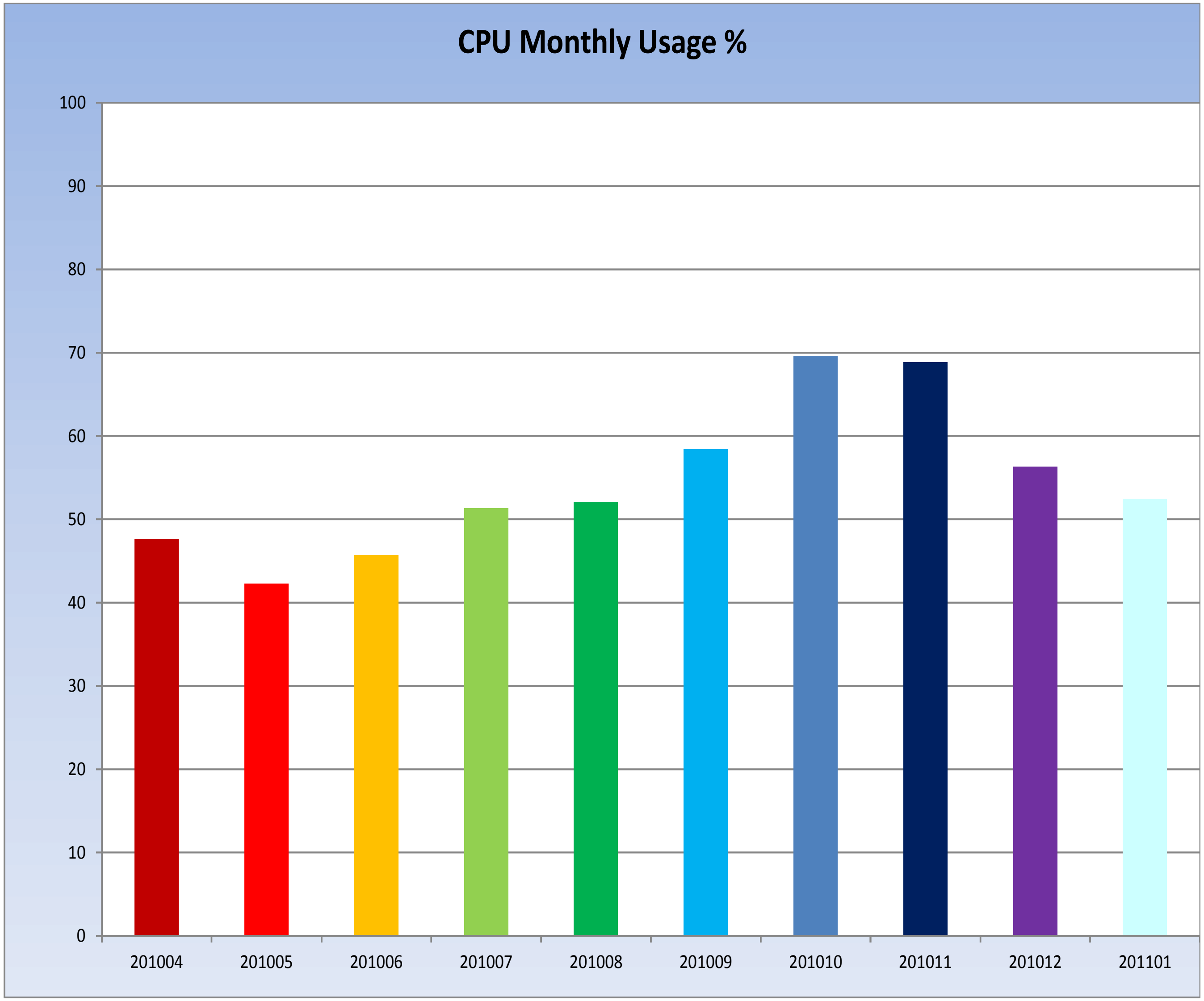
- **On the break record (c) Print to the output record using the alu the results of counter 1 divided by counter 3 divided by field q, (accumulated monthly cpu % divided by the amount of records divided by the amount of CPU's), reset counter 1 to zeroes. Then counters 11/3/field q (vio rate summary) , counters 13/3/field q (real I/O rate), number of records, reset counters to 0 appropriately. Pictures abound.**
- ***The contents of the print records in this slide is formatted to include the /,/ n on each line.***

Not presented 'cause not coded

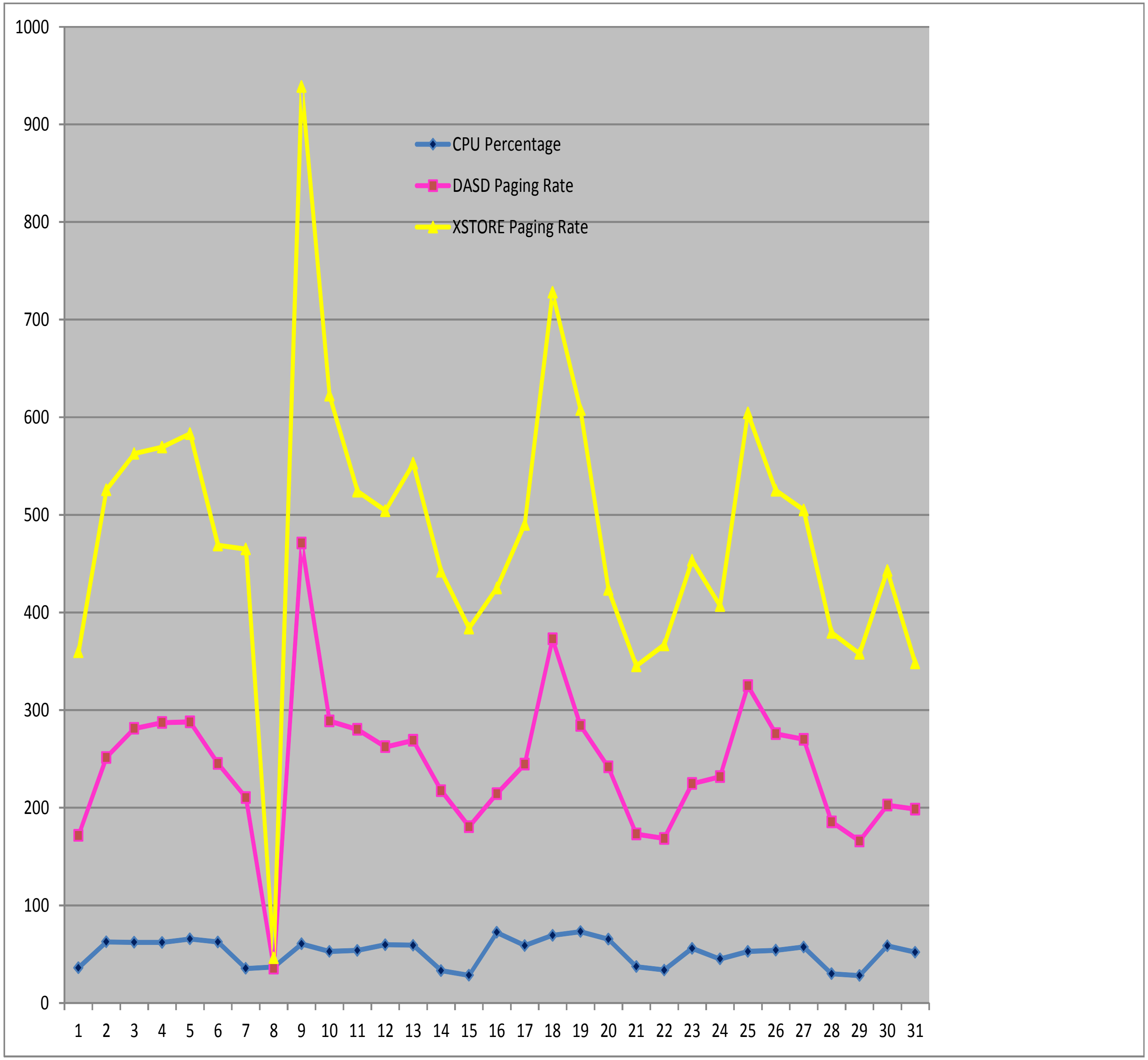
- Almost the full set of REXX functions may be spec'ed
- Boolean operations
- String processing
- Named fields – very cool especially with PERFKIT data.

Jury rigging refers to makeshift repairs or temporary contrivances, made with only the tools and materials that happen to be on hand. Originally a nautical term, on sailing ships a jury rig is a replacement mast and yards improvised in case of damage or loss of the original mast.

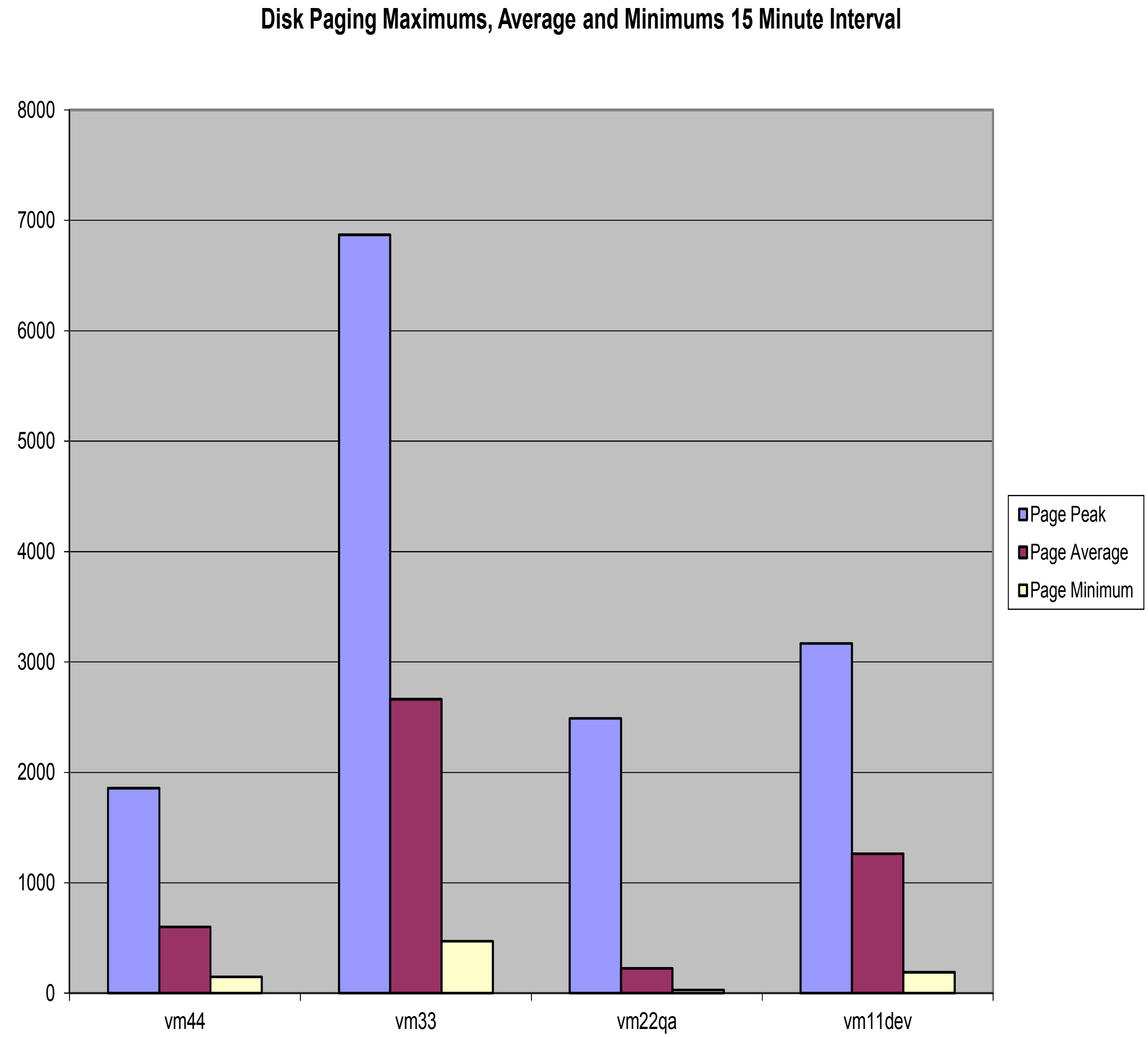
Let's see some jury riggging with SPECS and CSV data!



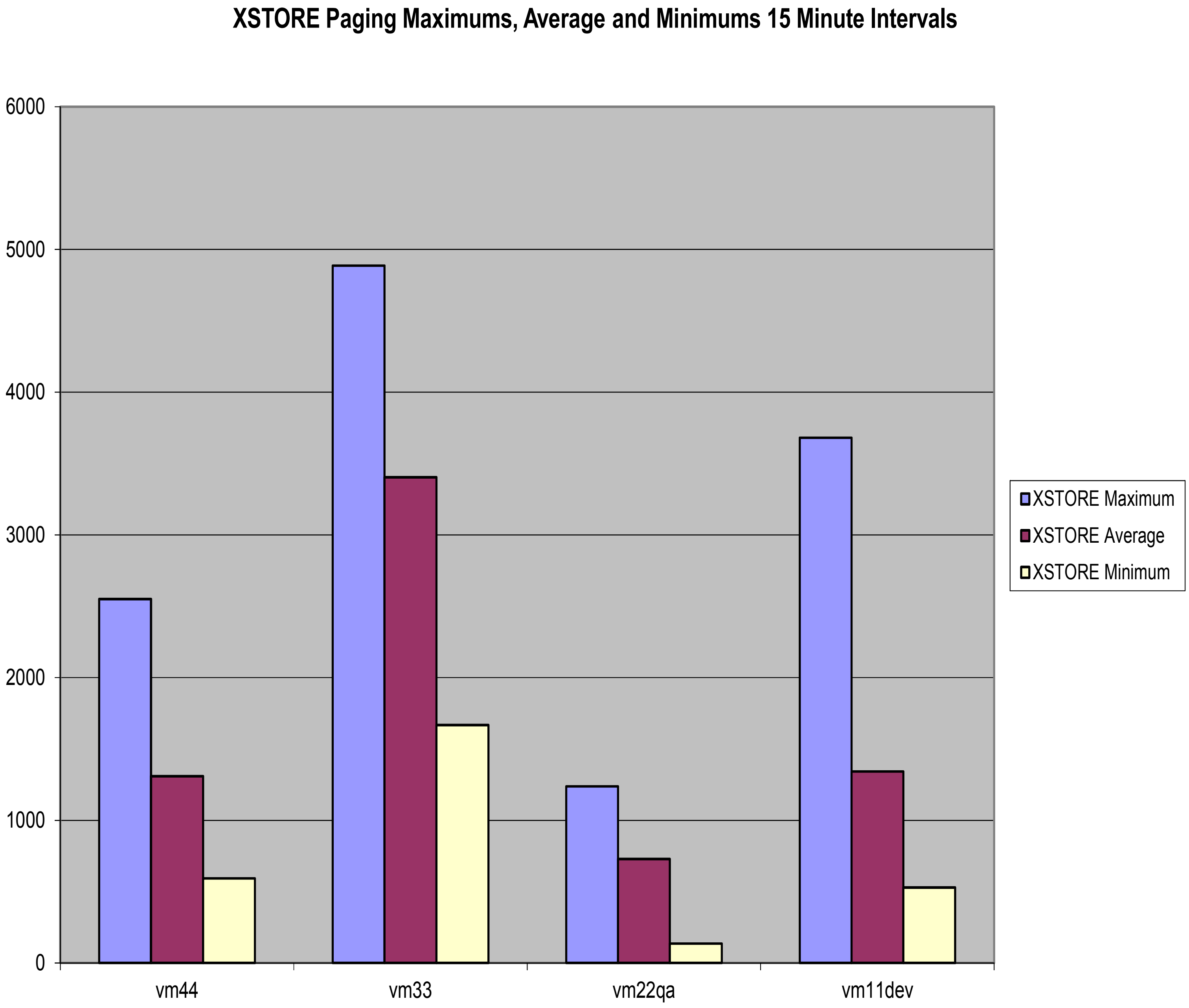
Use fields 11, and 10 – CPU busy % and number of IFLs), and jury rigged for charting. Key information for capacity planning.



Use fields 11, 10, 65 and 84 (CPU busy, # of IFLs, DASD page rates, XSTORE page rates rates) and jury rigged for charting.



Use field 65 (dasd page rates) and jury rigged for minimum average and maximum



Use field 84 (xstore rates) and jury rigged for minimum average and maximum

Velocity Data

- Velocity data produces CSV data as part of the product.
- Plugs in beautifully to the super spec'ing methods.
- No intermediate data transformation required.
- Used recently to process Linux data that was already in CSV format.
- Produced reports showing highest CPU consuming process ids, (PIDs), and program name.

About Vicom Infinity

- Account Presence Since Late 1990's
- IBM Premier Business Partner
- Reseller of IBM Hardware, Software, and Maintenance
- Vendor Source for the Last 4 Generations of Mainframes/IBM Storage
- Professional and IT Architectural Services

About VM RESOURCES LTD:

- Providing mainframe and Linux consulting and training since 1988
- Award winning consulting
- Complete set of z/VM and LoZ courses

Vicom Infinity and VM RESOURCES – a great team for all your mainframe needs!