Performance Tuning with DataStage Parallel Jobs

IBM Support Technical Exchange

Steve Wilkos
July 2015

Agenda

- Best practices for developing DataStage parallel jobs
- How to read the APT_DUMP_SCORE report
- Using NMON to analyze AIX and Linux performance
- Parallel job performance issues and how to resolve those issues
- References

Best Practices
for
Developing DataStage Parallel Jobs

Reduce processes generated at runtime

- Use a single-node configuration file (APT_CONFIG_FILE)

```
{
node “node1″
{
fastname “test"
pools “”
resource disk “C:/IBM/InformationServer/Server/Datasets/Node1″ {pools “”}
resource scratchdisk “C:/IBM/InformationServer/Server/Scratch/Node1″ {pools “}
}
```

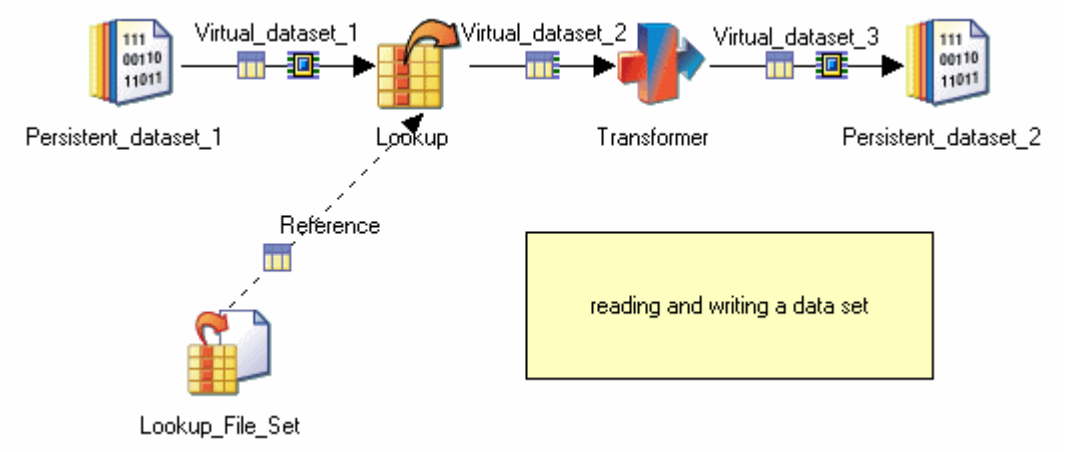
- Remove all partitioners and collectors (especially when using a single-node configuration file)
- Enable runtime column propagation on Copy stages with only one input and one output
- Minimize join structures (any stage with more than one input, such as Join, Lookup, Merge, Funnel)
- Minimize non-combinable stages such as Join, Aggregator, Remove Duplicates, Merge, Funnel, DB2 Enterprise, Oracle Enterprise, ODBC Enterprise, BuildOps, BufferOp
- Selectively (being careful to avoid deadlocks) disable buffering

Parallel datasets

- Use parallel datasets to land intermediate results between parallel jobs

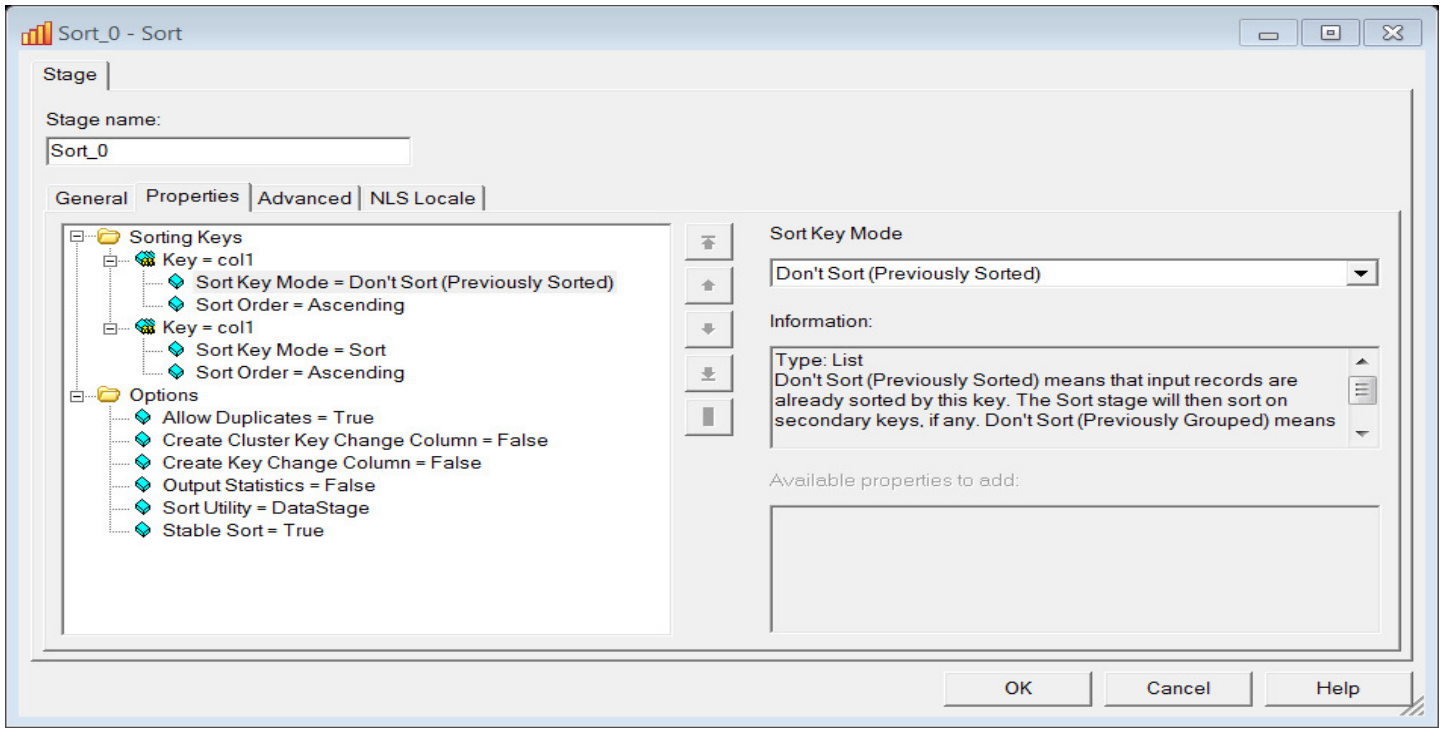
– Parallel datasets retain data partitioning and sort order

– Datasets can only be read by other DataStage parallel jobs



Sorting

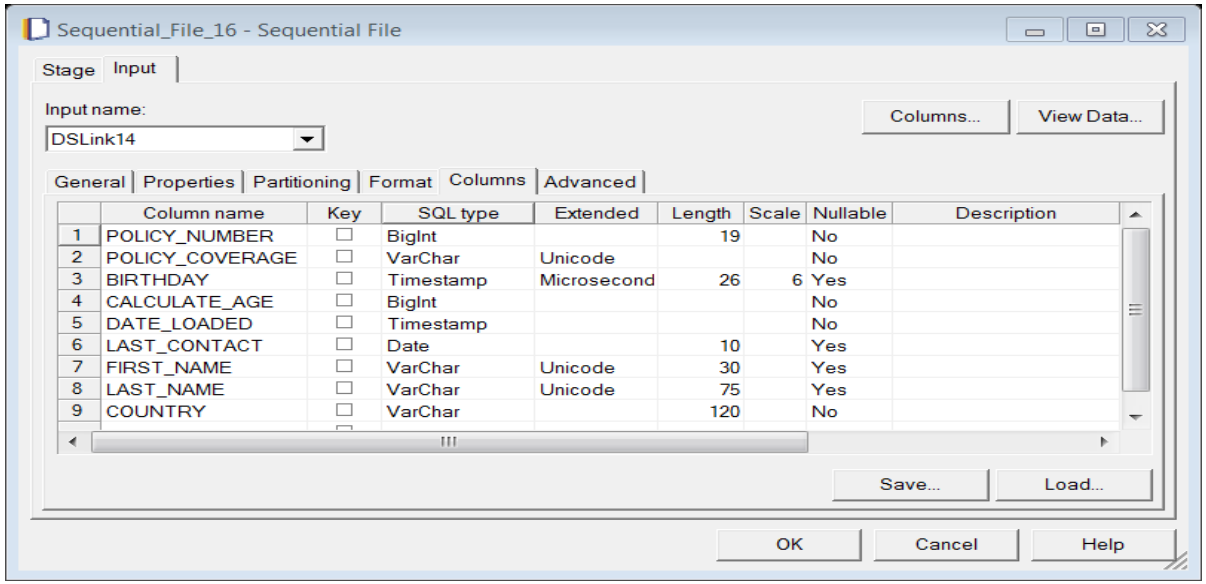
- Specify the **Don't Sort (Previously Sorted)** option for those key columns in the Sort stage if data has already been partitioned and sorted on a set of key columns



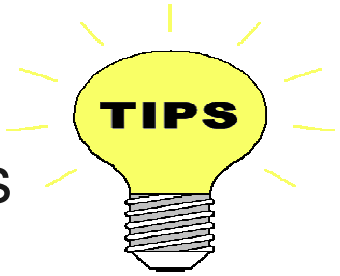
- When **writing** to parallel datasets sort order and partitioning are preserved.
- When **reading** from these datasets try to maintain this sorting, if possible, by using SAME partitioning
- Performance of individual sorts can be improved by increasing the memory usage per partition using the **Restrict Memory Usage (MB)** option of the standalone Sort stage.

Columns

- Remove unneeded columns as early as possible in the data flow
- When reading from database sources use a select list to read needed columns instead of the entire table
- When using runtime column propagation (RCP) it might be necessary to disable RCP for a particular stage to ensure that columns are actually removed using that stage's Output Mapping
- Specify a maximum length for Varchar columns
- Varchars without a maximum length (unbounded strings) can have a significant negative performance impact on a job flow



Performance tips



- test DataStage parallel jobs with a parallel configuration file ($APT_CONFIG_FILE) that has two or more nodes in its default pool
- more nodes does not always mean better job performance
- if you are seeing job startup time gradually increase usually that means that system resources are not available, which are critical to a job's performance
- reduce the number of transformers because transformers are individual shared object files
- a small fetch of 10-100 records could be faster if using lookup filesets rather than connecting to the database
- avoid using network drives for temporary directories
- avoid using GPFS on Linux for binaries and resource disks

How to read the APT_DUMP_SCORE report

The APT_DUMP_SCORE report

- The APT_DUMP_SCORE report records activity within the Information Server parallel engine
- Set the environment variable APT_DUMP_SCORE=TRUE
- In DataStage Administrator client click Parallel -> Reporting
- When a DataStage job is executed the data flow information in the compile job is combined with the APT_CONFIG_FILE configuration file to produce a detailed execution plan named the score
- The score is useful in analyzing job performance

The APT_DUMP_SCORE report

- The dump score contains two sections -- the data sets (DS) and the operators (OP).

Data sets - The data sets that are listed in the score are the same type of data sets that you create with the Data Set stage.

In this report they are temporary memory and/or disk storage during the job's run

Operators - Operators are individual parallel engine stages that you might see on the user interface

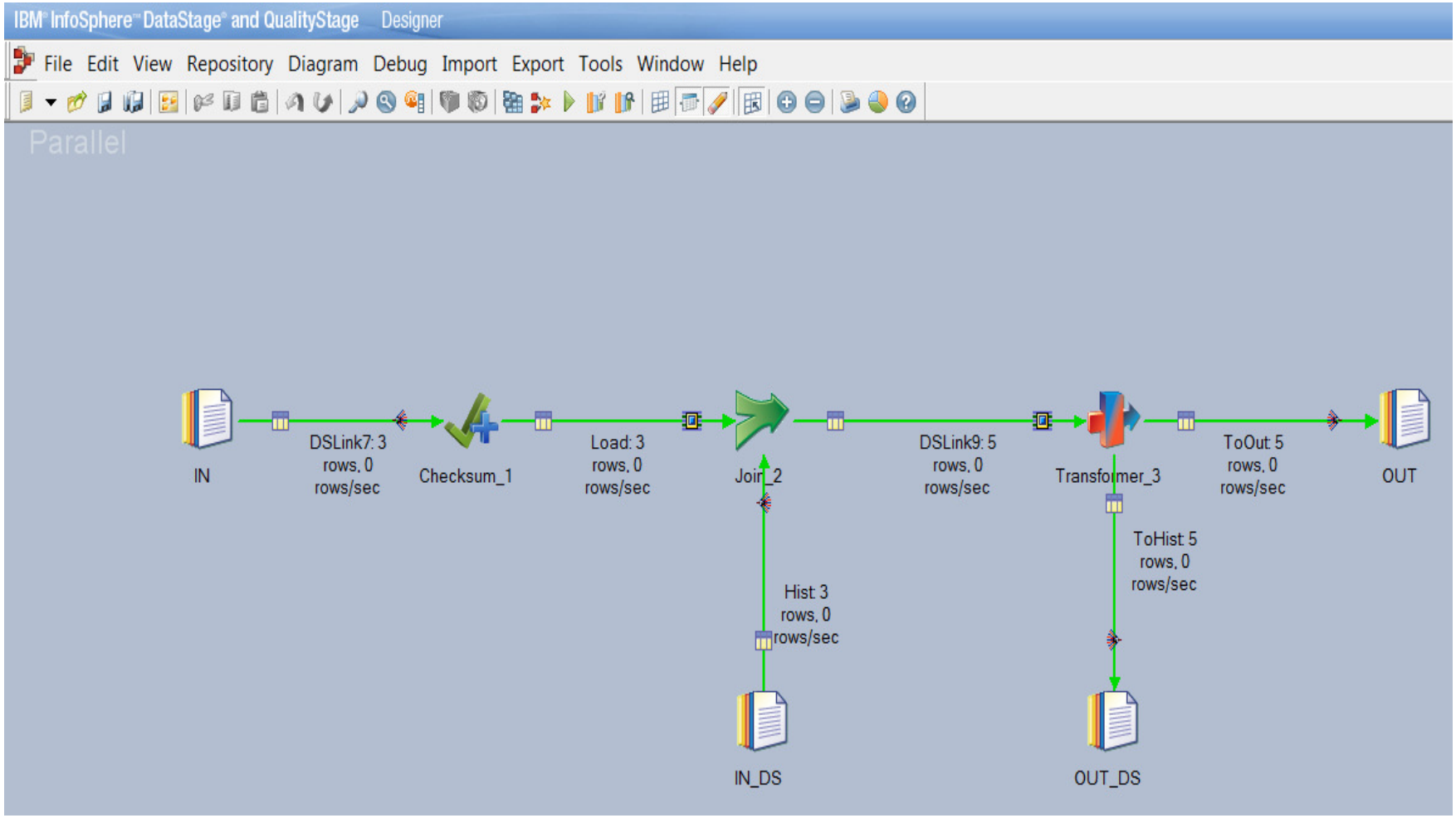
APT_CONFIG_FILE file

- The APT_CONFIG_FILE file specifies the nature and amount of parallelism along with the specific resource that are used to run a job
- Default location is /IBM/InformationServer/Server/Configurations

```
{
node "node1"
        {
                fastname “development"
                pools ""
                resource disk "/u1/IS113/IBM/InformationServer/Server/Datasets" {pools ""}
                resource scratchdisk "/u1/IS85/IBM/InformationServer/Server/Scratch" {pools ""}
        }
node "node2"
        {
                fastname “development"
                pools ""
                resource disk "/u1/IS113/IBM/InformationServer/Server/Datasets" {pools ""}
                resource scratchdisk "/u1/IS85/IBM/InformationServer/Server/Scratch" {pools ""}
        }

}
```

Sample DataStage parallel job



APT_DUMP_SCORE report dataset section

```
 Message: main_program: This step has 8 datasets:
ds0: {op0[1p] (sequential IN)
      eAny<>eCollectAny
      op1[2p] (parallel Checksum_1)}
ds1: {op1[2p] (parallel Checksum_1)
      eOther(APT_HashPartitioner { key={ value=SK }
})#>eCollectAny
      op3[2p] (parallel inserted tsort operator {key={value=SK, subArgs={asc}}}(0) in Join_2)}
ds2: {op2[1p] (sequential IN_DS)
      eOther(APT_HashPartitioner { key={ value=SK }
})<>eCollectAny
      op4[2p] (parallel inserted tsort operator {key={value=SK, subArgs={asc}}}(1) in Join_2)}
ds3: {op3[2p] (parallel inserted tsort operator {key={value=SK, subArgs={asc}}}(0) in Join_2)
      [pp] eSame=>eCollectAny
      op5[2p] (parallel APT_JoinSubOperatorNC in Join_2)}
ds4: {op4[2p] (parallel inserted tsort operator {key={value=SK, subArgs={asc}}}(1) in Join_2)
      [pp] eSame=>eCollectAny
      op5[2p] (parallel APT_JoinSubOperatorNC in Join_2)}
ds5: {op5[2p] (parallel APT_JoinSubOperatorNC in Join_2)
      eAny=>eCollectAny
      op6[2p] (parallel APT_TransformOperatorImplV0S3_FullOuter_Join_Sample_Transformer_3 in
Transformer_3)}
ds6: {op6[2p] (parallel APT_TransformOperatorImplV0S3_FullOuter_Join_Sample_Transformer_3 in
Transformer_3)
      >>eCollectAny
      op7[1p] (sequential APT_RealFileExportOperator1 in OUT_DS)}
ds7: {op6[2p] (parallel APT_TransformOperatorImplV0S3_FullOuter_Join_Sample_Transformer_3 in
Transformer_3)
      >>eCollectAny
      op8[1p] (sequential APT_RealFileExportOperator1 in OUT)}
```


Data sets structure:

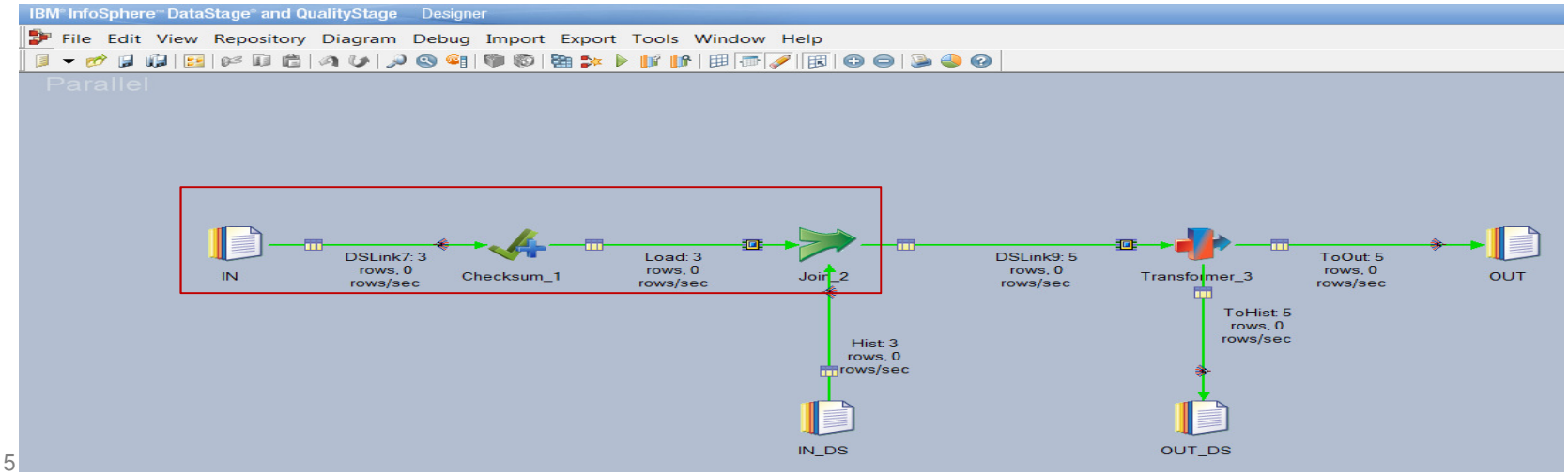
```
ds0: {op0[1p] (sequential IN)
      eAny<>eCollectAny
      op1[2p] (parallel Checksum_1)}
ds1: {op1[2p] (parallel Checksum_1)
      eOther(APT_HashPartitioner { key={ value=SK }
})#>eCollectAny
      op3[2p] (parallel inserted tsort operator {key={value=SK, subArgs={asc}}}(0) in
Join_2)}
```

- The name of the dataset is ds0. Within the curly brackets there are three stages:

  -- the source of the data set is operator 0, sequential IN

  -- the activity of the data set is operator 1, parallel Checksum_1

  -- the target of the data set is operator 3, parallel inserted tsort operator in Join_2



Data Set symbols

- The symbols between the originating partitioning method and the target read method translates to the parallelism of the partitioning.

- Here is the list of the symbols and their definition:

  -> Sequential to Sequential
  <> Sequential to Parallel
  => Parallel to Parallel (*SAME*)
  #> Parallel to Parallel (*NOT SAME*)
  >> Parallel to Sequential
  > No Source or Target

- eOther is the originating or input method

- eCollectAny is the target read method

```
ds0: {op0[1p] (sequential IN)
      eAny<>eCollectAny
      op1[2p] (parallel Checksum_1)}
ds1: {op1[2p] (parallel Checksum_1)
      eOther(APT_HashPartitioner { key={ value=SK }
})#>eCollectAny
      op3[2p] (parallel inserted tsort operator {key={value=SK, subArgs={asc}}}(0) in Join_2)}
```

APT_DUMP_SCORE report operator section

```
It has 9 operators:
op0[1p] {(sequential IN)
    on nodes (
      node1[op0,p0]
    )}
op1[2p] {(parallel Checksum_1)
    on nodes (
      node1[op1,p0]
      node2[op1,p1]
    )}
op2[1p] {(sequential IN_DS)
    on nodes (
      node1[op2,p0]
    )}
op3[2p] {(parallel inserted tsort operator {key={value=SK, subArgs={asc}}}(0) in Join_2)
    on nodes (
      node1[op3,p0]
      node2[op3,p1]
    )}
op4[2p] {(parallel inserted tsort operator {key={value=SK, subArgs={asc}}}(1) in Join_2)
    on nodes (
      node1[op4,p0]
      node2[op4,p1]
    )}
op5[2p] {(parallel APT_JoinSubOperatorNC in Join_2)
    on nodes (
      node1[op5,p0]
      node2[op5,p1]
    )}
op6[2p] {(parallel APT_TransformOperatorImplV0S3_FullOuter_Join_Sample_Transformer_3 in Transformer_3)
    on nodes (
      node1[op6,p0]
      node2[op6,p1]
    )}
op7[1p] {(sequential APT_RealFileExportOperator1 in OUT_DS)
    on nodes (
      node2[op7,p0]
    )}
op8[1p] {(sequential APT_RealFileExportOperator1 in OUT)
    on nodes (
      node2[op8,p0]
    )}
```


Operators structure

```
op0[1p] {(sequential IN)
    on nodes (
      node1[op0,p0]
    )}
op1[2p] {(parallel Checksum_1)
    on nodes (
      node1[op1,p0]
      node2[op1,p1]
    )}
```

- The two operators are op0 and op1
- op0[1p] indicates that one partition is provided to the operator by the engine
- op1[2p] indicates that two partitions are provided to the operator by the engine
- (sequential IN) indicates execution mode sequential and stage name IN
- (parallel Checksum_1) indicates execution mode parallel and stage name Checksum_1
- op0 runs on node 1
- op1 runs are node 1 and node 2

Buffer operators

- Buffer operators are used when the downstream operator is at risk of getting overloaded with data while it is processing

```
op6[4p] {(parallel buffer(0))
on nodes (
node1[op6,p0]
node2[op6,p1]
```

- Buffer operators are an attempt to produce a buffer zone where two things happen:

1) The buffer operator communicates with the upstream operator to manage the sending of data

2) The buffer operator holds on to the data until the downstream operator is ready for the next block of data

- If your parallel job is running slowly look for the number of buffer operators in the report
- Buffer operators prevent race conditions between operators
- Best practices job design can reduce the amount of buffering that occurs

The APT_DUMP_SCORE report

To improve parallel job performance:

- If your parallel job is running slowly look for the number of buffer operators in the report
- Examine the report for sort operators because sorting can be detrimental to parallel job performance
- Join stages require data to be sorted so the engine inserts a tsort operator
- Data sets use memory which can impact job performance
- Ensure that your intended design is correctly implemented by the parallel engine

Using NMON
to analyze AIX and Linux performance

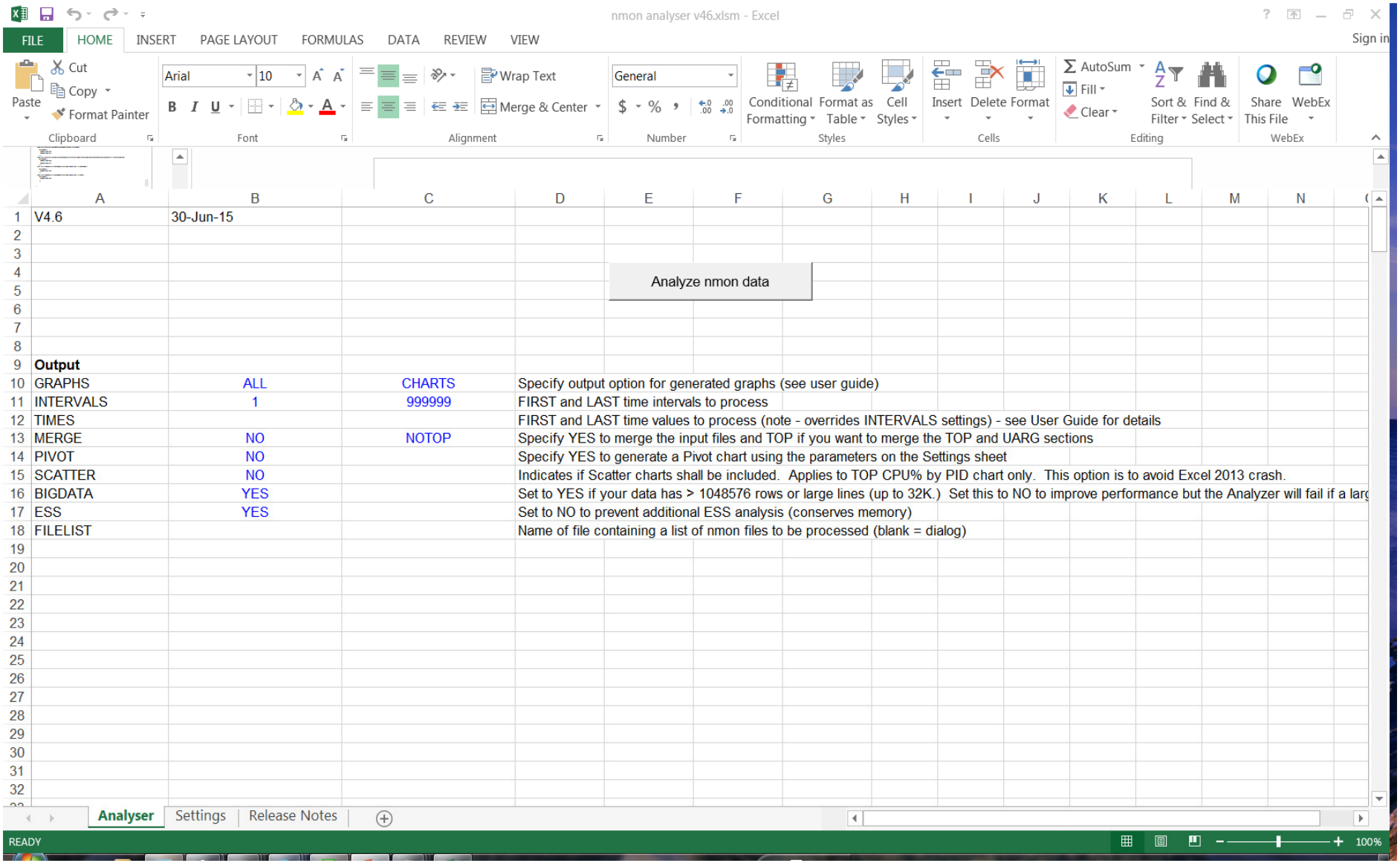
nmon

You can download the nmon analyzer here: (latest version is 4.6 nmon_analyser_v46.zip)

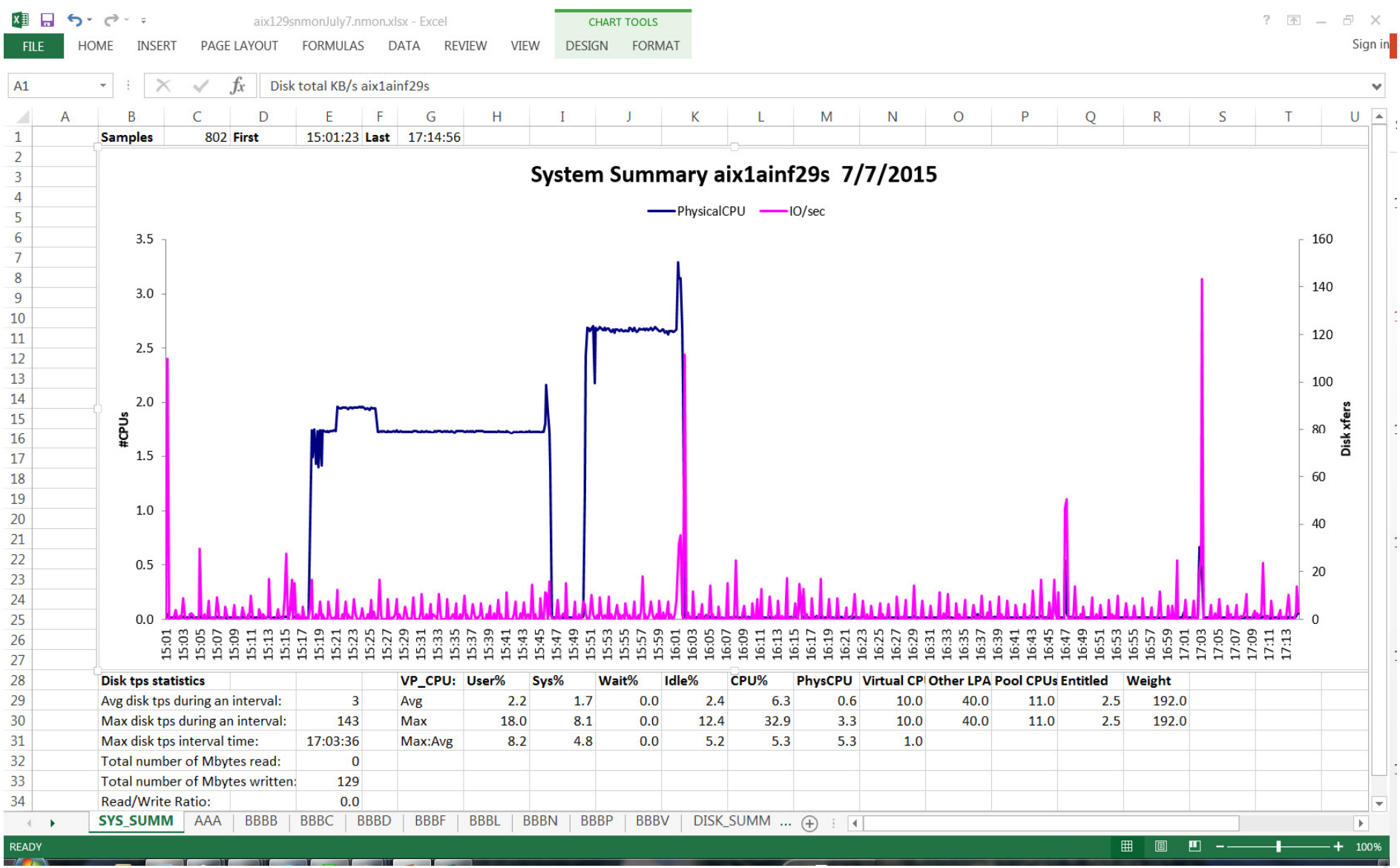
https://www.ibm.com/developerworks/community/wikis/home?lang=en#!/wiki/Power%20Systems/page/nmon_analyser

- nmon creates graphs to aid in analyzing performance issues
- nmon works with Microsoft Excel 2003 and later on AIX and Linux
- nmon is a free tool that is not supported by IBM

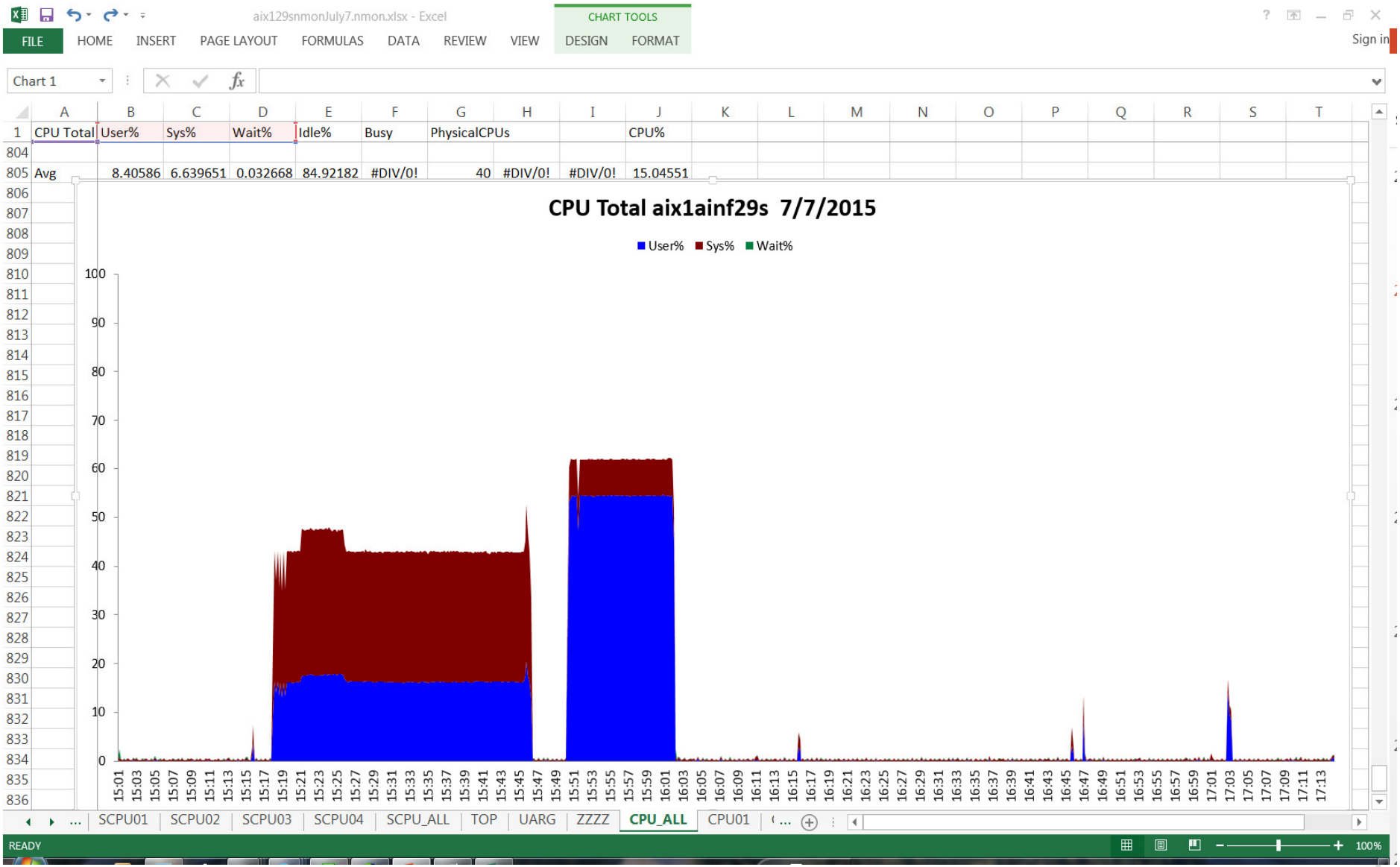
click Analyze nmon data



The excel file that is created

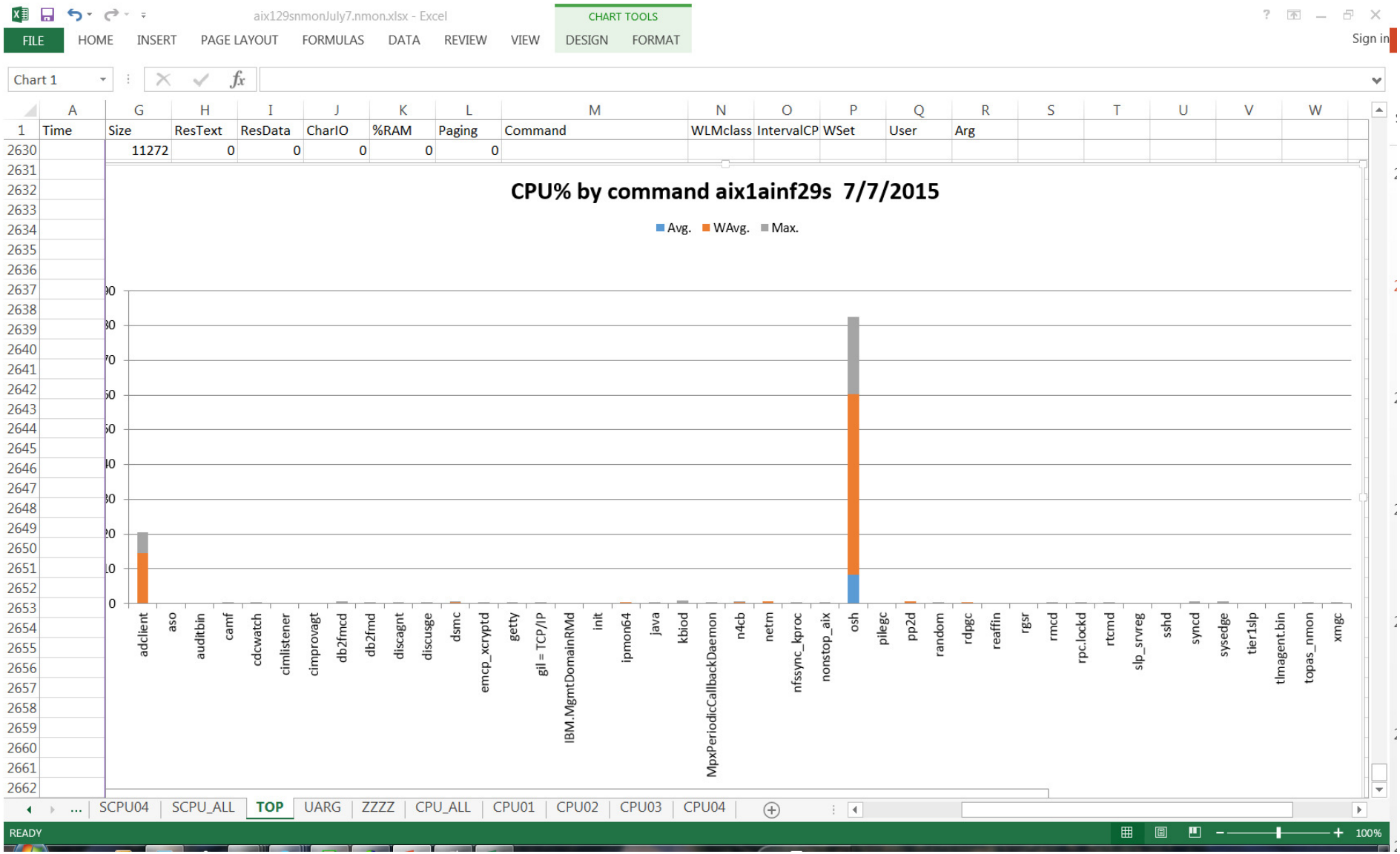


CPU utilization

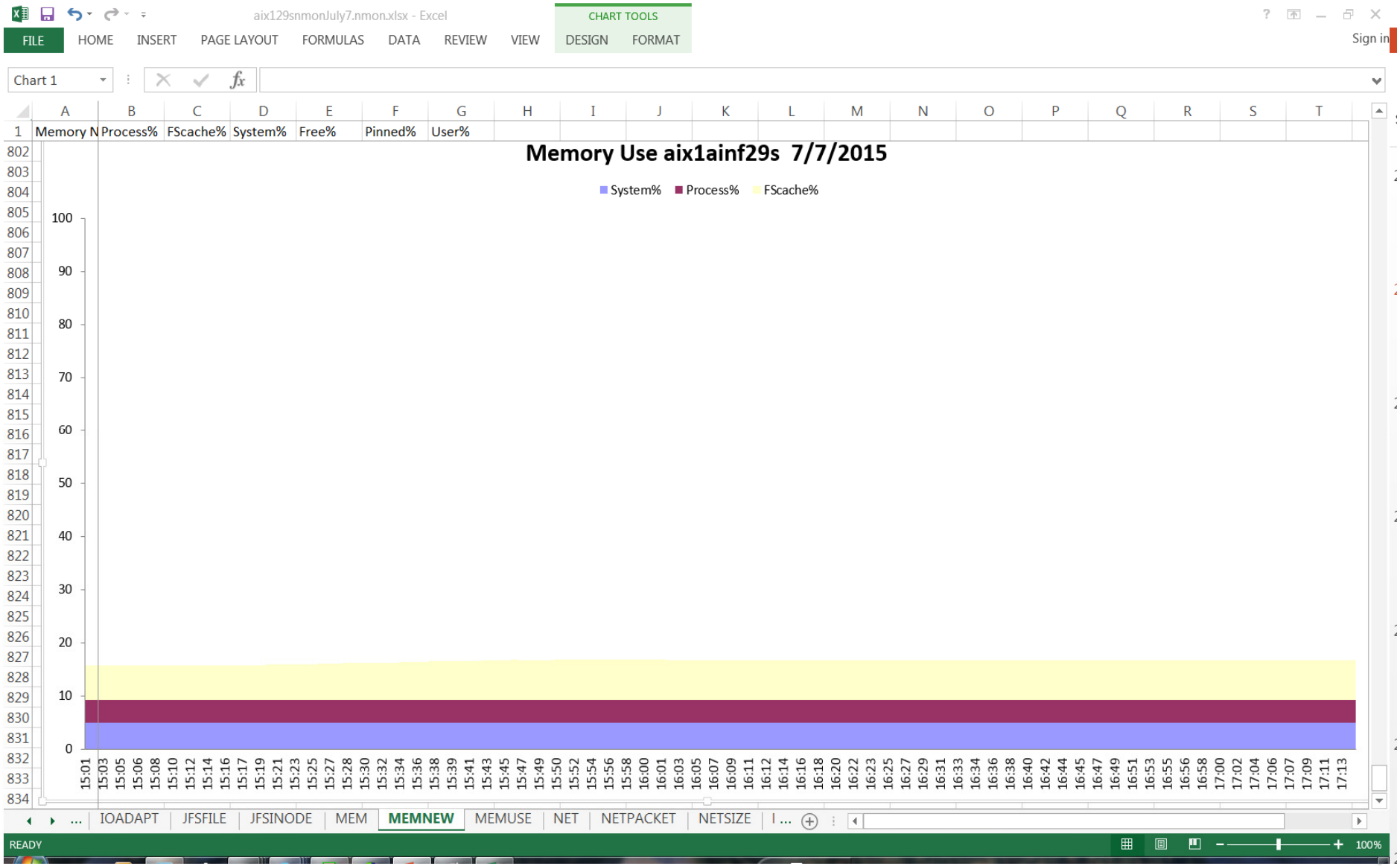


CPU Total	User%	Sys%	Wait%	Idle%	Busy	PhysicalCPUs			CPU%
Avg	8.40586	6.639651	0.032668	84.92182	#DIV/0!	40	#DIV/0!	#DIV/0!	15.04551

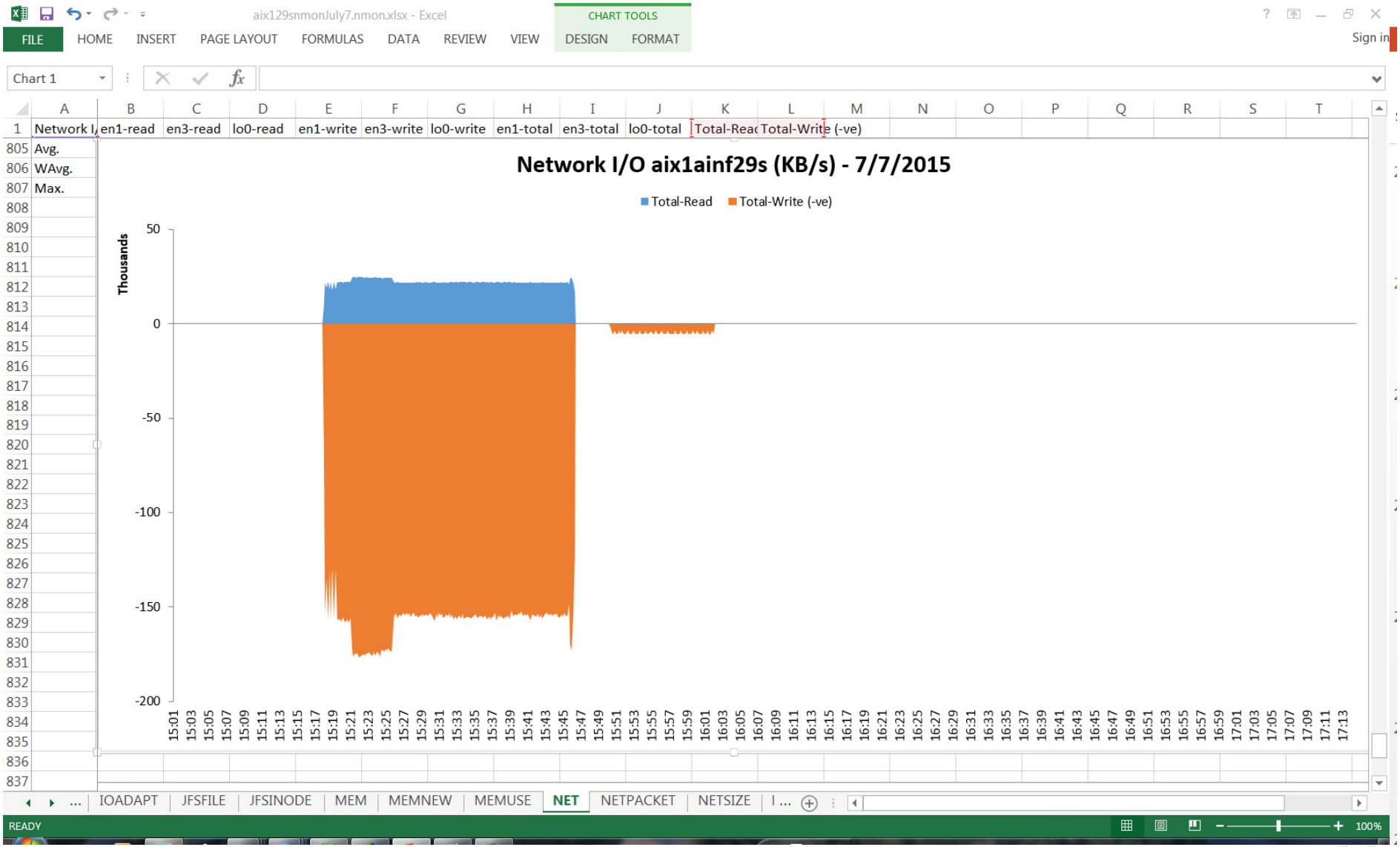
Analyze CPU% by command



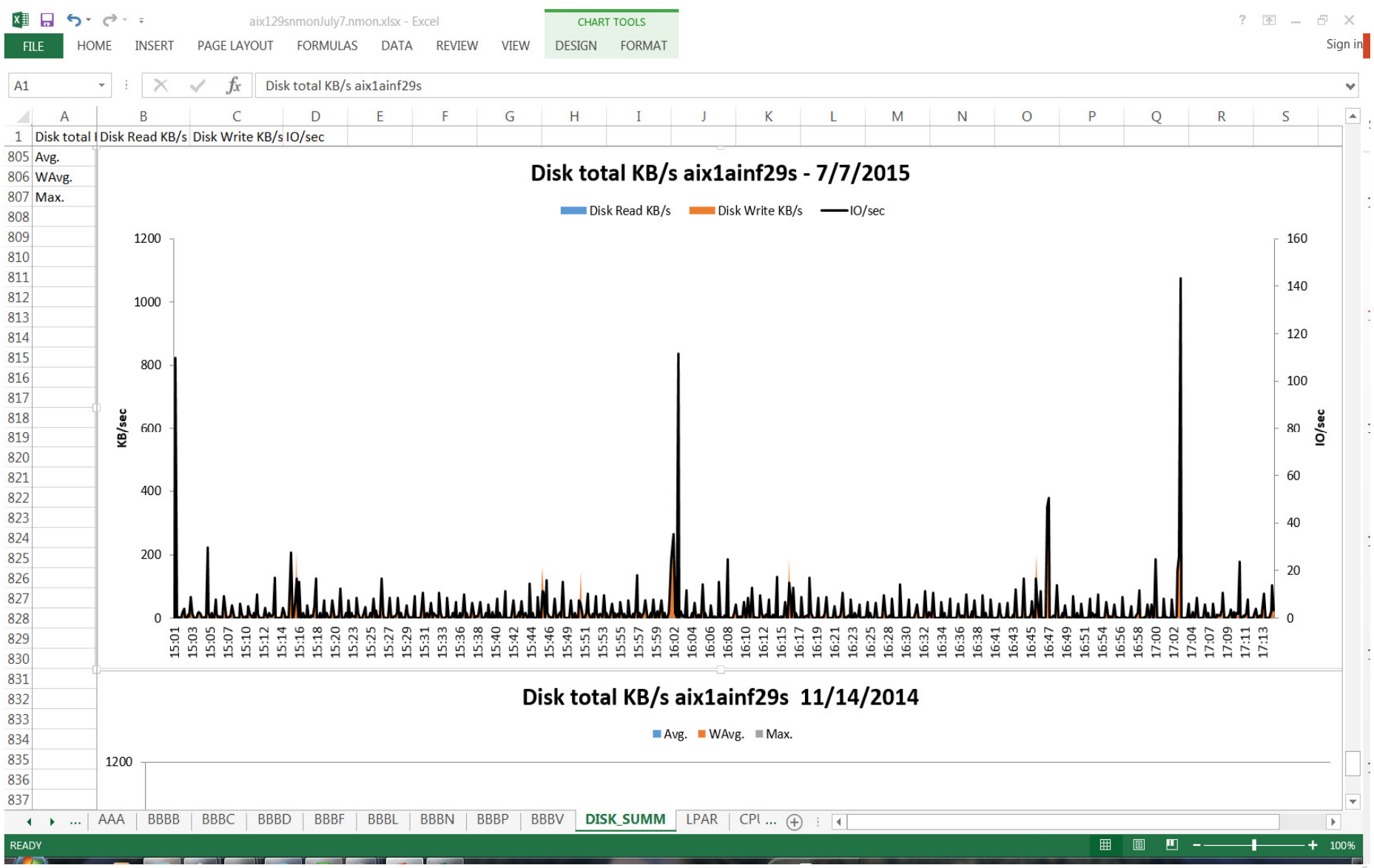
Memory usage



Network I/O


aix129snmonJuly7.nmon.xlsx - Excel
CHART TOOLS
FILE
HOME
INSERT
PAGE LAYOUT
FORMULAS
DATA
REVIEW
VIEW
DESIGN
FORMAT
Sign in
Chart 1
Network I/O aix1ainf29s (KB/s) - 7/7/2015
Total-Read
Total-Write (-ve)
Thousands
50
0
-50
-100
-150
-200
15:01
15:03
15:05
15:07
15:09
15:11
15:13
15:15
15:17
15:19
15:21
15:23
15:25
15:27
15:29
15:31
15:33
15:35
15:37
15:39
15:41
15:43
15:45
15:47
15:49
15:51
15:53
15:55
15:57
15:59
16:01
16:03
16:05
16:07
16:09
16:11
16:13
16:15
16:17
16:19
16:21
16:23
16:25
16:27
16:29
16:31
16:33
16:35
16:37
16:39
16:41
16:43
16:45
16:47
16:49
16:51
16:53
16:55
16:57
16:59
17:01
17:03
17:05
17:07
17:09
17:11
17:13
Network I/
en1-read
en3-read
lo0-read
en1-write
en3-write
lo0-write
en1-total
en3-total
lo0-total
Total-Rea
Total-Write (-ve)
Avg.
WAvg.
Max.
IOADAPT
JFSFILE
JFSINODE
MEM
MEMNEW
MEMUSE
NET
NETPACKET
NETSIZE
READY
100%

Disk total KB/s



How the nmon report can help you improve job performance

- % CPU is high

  Acton: identify which processes are consuming large amounts of CPU

- Disk % Busy is high

  Action: cache more data in-memory

- Network I/O is high

  Action: increase network bandwidth

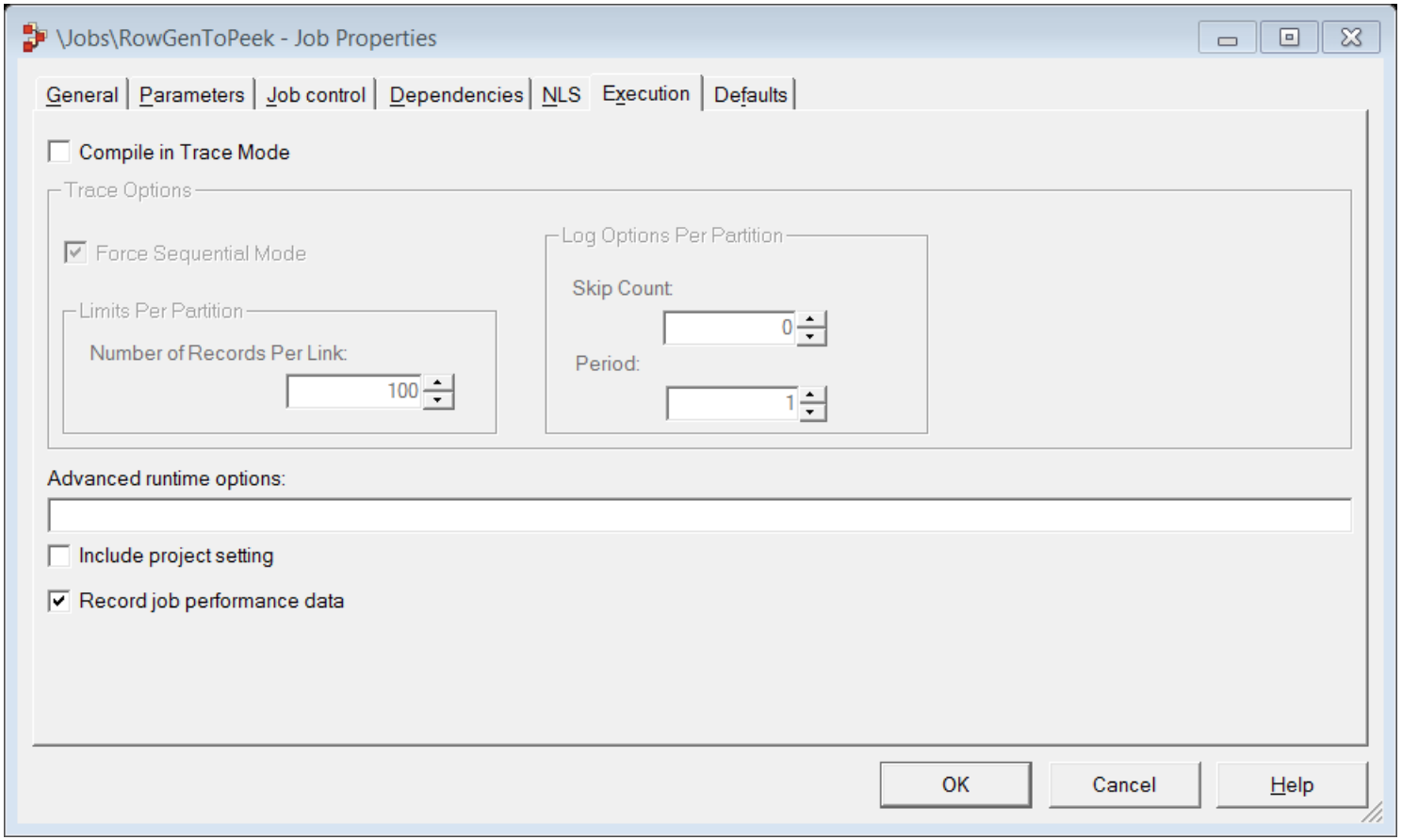
- Physical memory usage is high

  Action: identify which processes are using memory

Parallel job performance issues and how to resolve those issues

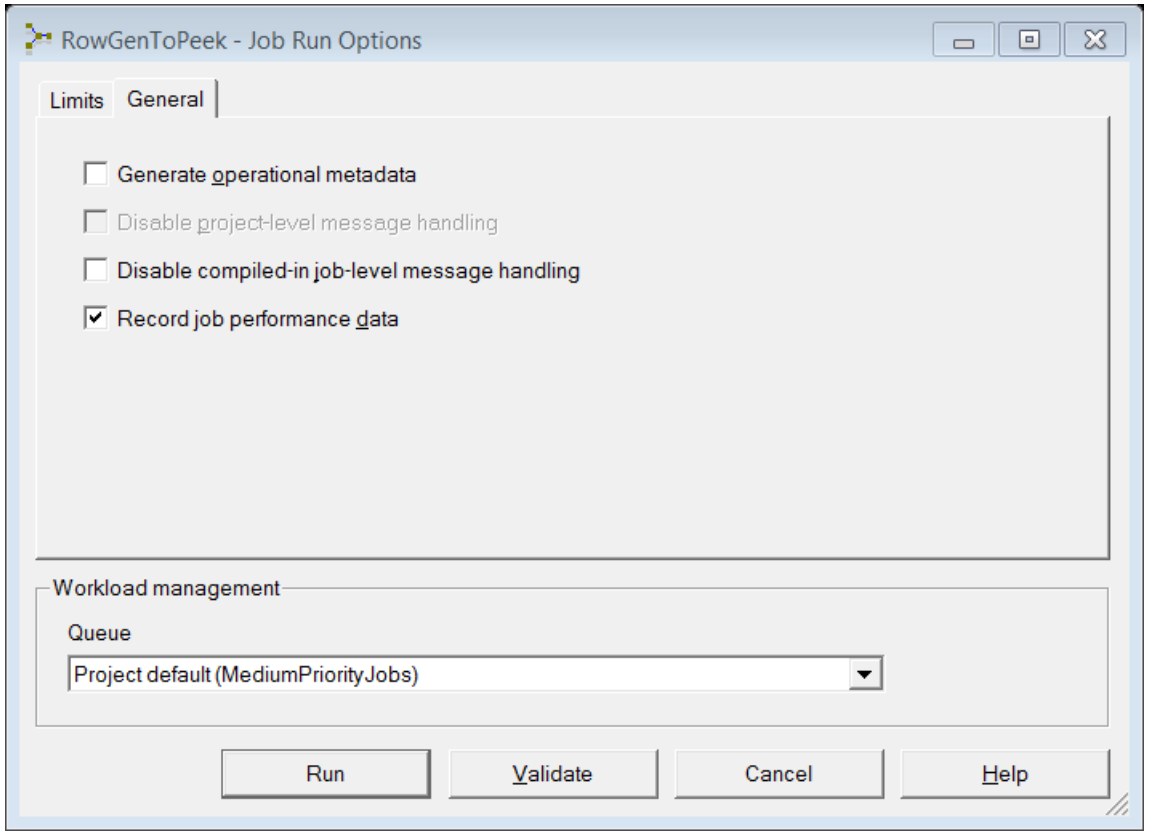
Recording performance data at design time

1. Open a job in the Designer client
2. Click Edit > Job Properties
3. Click the Execution page
4. Select the Record job performance data check box



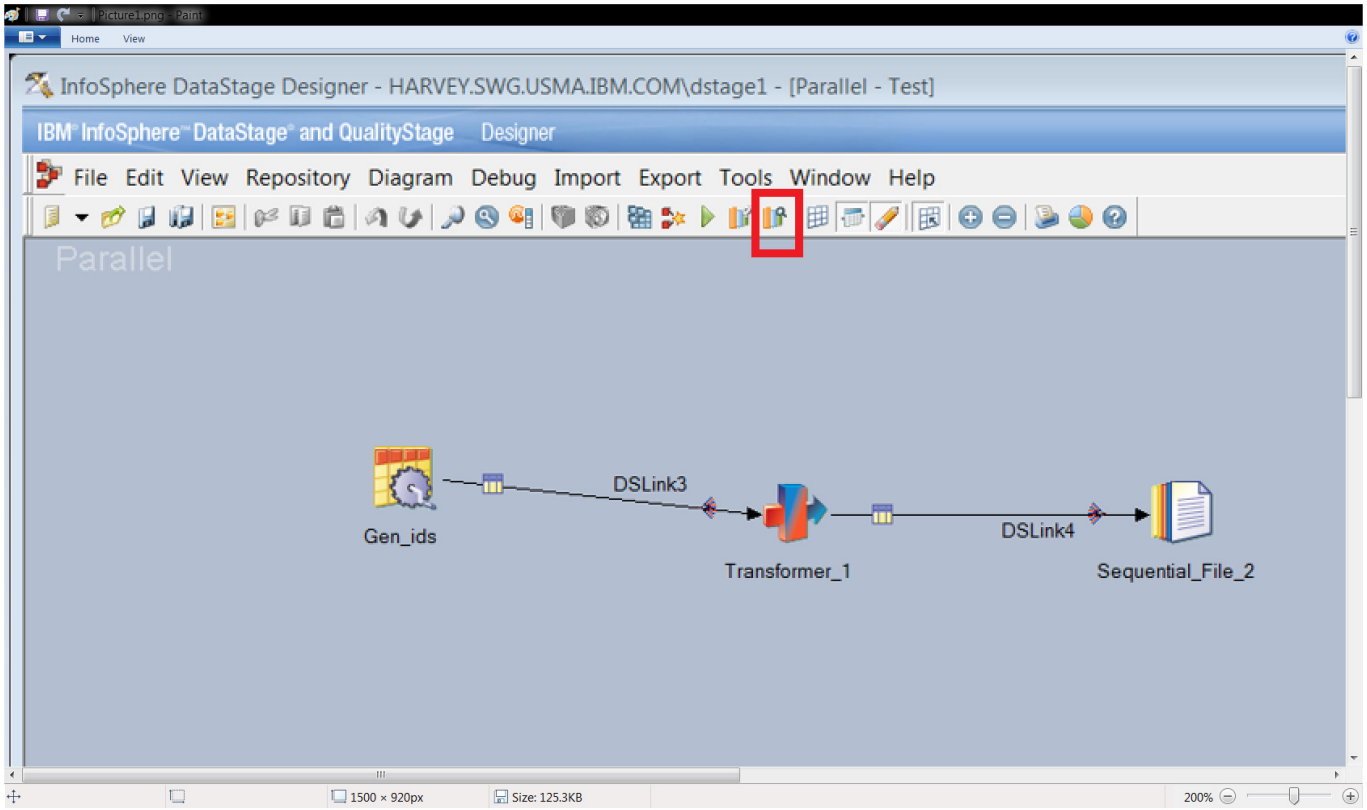
Recording performance data at run time

1. Open a job in the Designer client, or select a job in the display area of the Director client
2. Click the Run button on the toolbar to open the Job Run Options window
3. Click the General page
4. Select the Record job performance data check box

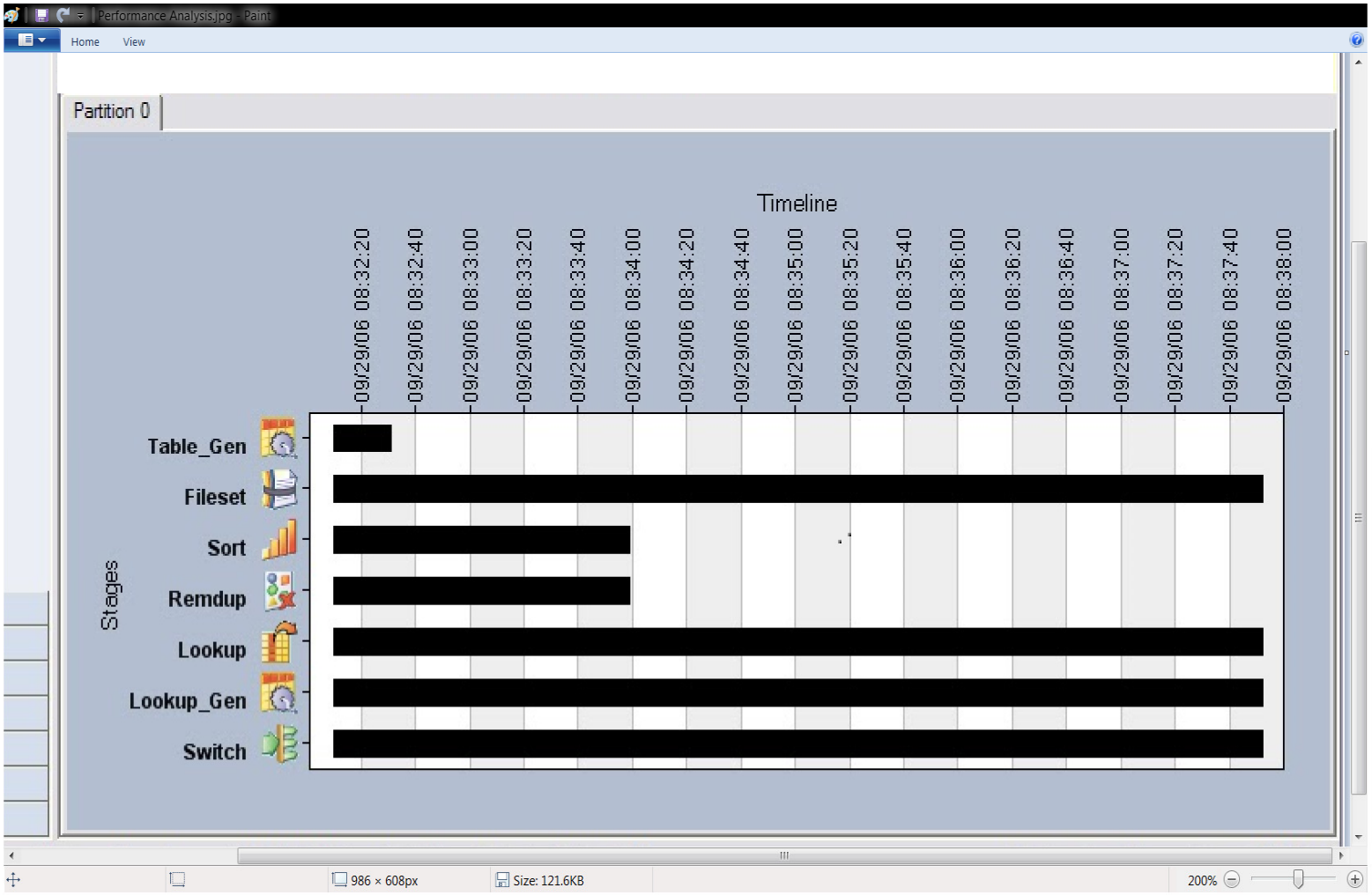


Viewing performance data

- Open the Performance Analysis window by using one of the following methods:
  - In the Designer client, click File > Performance Analysis
  - In the Director client, click Job > Analyze Performance
  - In either client, click the Performance Analysis toolbar button



Viewing performance data



Environment variables

- APT_CONFIG_FILE

  File specifies the nature and amount of parallelism along with the specific resources that are used to run a job

- APT_SCORE_DUMP

  Report records activity within the Information Server parallel engine

- APT_PM_PLAYER_TIMING

  Identify which stages in a job are consuming large amounts of CPU time and how much data the stage is processing

- APT_NO_SORT_INSERTION

  Prevents the automatic insertion of sort components in your job to optimize the performance of the operators in your data flow

- APT_BUFFER_MAXIMUM_MEMORY

  Sets the default value of Maximum memory buffer size.

Parallel job performance issue:

- On RedHat Enterprise Linux 6.1 operating system parallel jobs run with degradated performance
- On RedHat Linux 6.1 system there is a 10 - 90% performance degradation depending on the hardware configuration and DataStage node configuration during parallel job runs. An increase in the number of CPU cores and DataStage node counts used on the system leads to more performance degradation
- This is caused by parallel jobs using transformer stages and use of the CurrentTimestampMS() function in the transformer stage on the Redhat Linux 6.1 system
- On the computer that has the Redhat Enterprise Linux 6.1 operating system installed, set the environment variable as follows:

  TZ=:/etc/localtime

  before running parallel jobs. The CPU time improves

- www.ibm.com/support/docview.wss?uid=swg21598208

Parallel job performance issue:

- DataStage parallel job has slow performance sending data to remote compute nodes
- When a DataStage job is setup with nodes running on multiple physical machines the transmission of data for large VarChar columns to remote nodes may be very slow compared to sending the same data to local nodes
- DataStage can transfer data faster to local nodes versus remote nodes
- When sending VarChar data to remote nodes if a length for the data has been declared, then space is allocated for the maximum size and the full bound size is transmitted to the remote node.
- However, if you have unbound data size (length of VarChar is undefined), then the size of data sent to the remote system is the actual variable size of each record
- The difference will not usually be that large unless the max size of VarChar column is much larger than the average size.
- www.ibm.com/support/docview.wss?uid=swg21667407

Parallel job performance issue:

- After upgrading to RedHat Linux 6 parallel jobs are using more memory
- The increased memory usage is caused by the Linux kernel ‘Huge Page’ feature
- This feature has been in Linux for years but normally was not enabled
- Starting with Red Hat Enterprise Linux 6 the Huge Page feature is enabled by default ("Transparent Huge Pages")
- Users will experience 30% or more increase in memory usage during parallel job execution
- Resolution is to turn off Huge Pages
- Documented in technote:

  http://www.ibm.com/support/docview.wss?uid=swg21664196

References

References

Best practices for developing DataStage parallel jobs

Developing DataStage and QualityStage parallel jobs
http://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/com.ibm.swg.im.iis.ds.nav.doc/topics/dshold_developing_parallel_ds_and_qs_jobs.html

Designing parallel jobs
http://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/g_deeref_Parallel_Jobs_General_Information.html?lang=en

InfoSphere DataStage Parallel Framework Standard Practices
http://www.redbooks.ibm.com/redbooks/pdfs/sg247830.pdf

References

How to read the APT_DUMP_SCORE report

InfoSphere DataStage parallel jobs: Understanding the content of the APT_DUMP_SCORE report
http://www.ibm.com/support/docview.wss?uid=swg21595704

Using NMON to analyze AIX and Linux performance

nmon performance: A free tool to analyze AIX and Linux performance
http://www.ibm.com/developerworks/aix/library/au-analyze_aix/

References

Common parallel job performance issues and how to resolve those issues

Running parallel jobs in debug mode
http://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/debuggingparalleljobsusingthedebugger.html?lang=en

Debugging parallel jobs with the debugging stages
http://www.ibm.com/support/knowledgecenter/SSZJPZ_11.3.0/com.ibm.swg.im.iis.ds.parjob.dev.doc/topics/debuggingyoudesigns.html?lang=en

How to turn on Tracing for DataStage Parallel Job
http://www.ibm.com/support/docview.wss?uid=swg21441558

InfoSphere DataStage: Parallel Job Performance Issue on Redhat Linux 6.1 System
http://www.ibm.com/support/docview.wss?uid=swg21598208

DataStage parallel job job has slow performance sending data to remote compute nodes
http://www.ibm.com/support/docview.wss?uid=swg21667407

Thank-you!