IBM i Performance Analysis with Performance Data Investigator Tool

Finding answers to 5 basic performance questions

Prepared By : Satid Singkorapoom

Power Systems and IBM i Consultant

April 2022

IBM i Performance Data Investigator (PDI) tool

A built-in IBM i performance report tool that produces graphical performance data charts that accommodate uncomplicated interpretation on performance health of various components of Power servers running IBM i.

A picture is worth a thousand words.

Do we need to add more CPU core?

Do we need to add more memory?

I have multiple disk pools (ASP).
How does each perform?

Do we have workload growth or reduction?

Does performance tuning work?

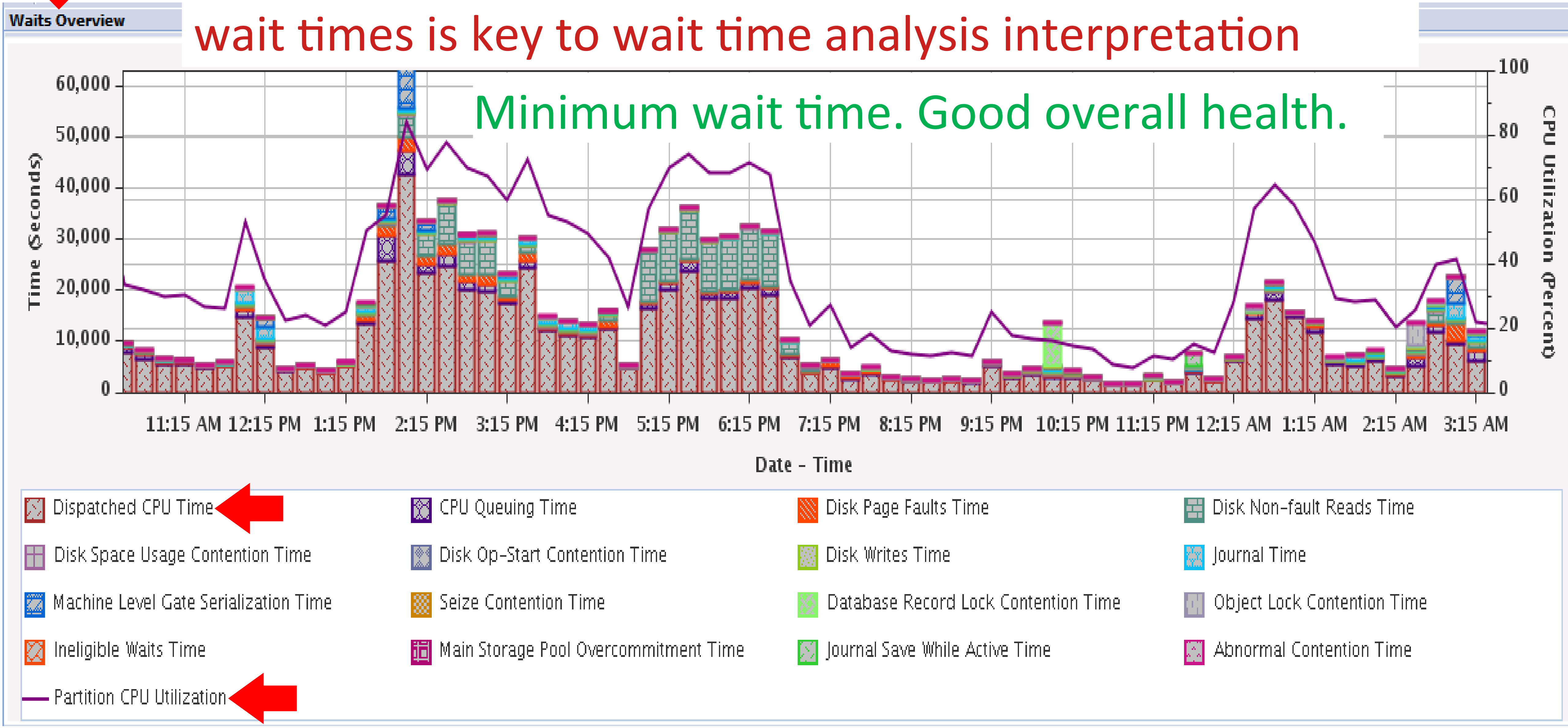
Do we need to add more CPU core?

Use PDI charts on Wait Overview and Wait by (Generic) Job or Task and Wait by Subsystem.

Dispatched CPU Time is the most desirable component in these charts that any active jobs need.

More CPU power is needed when CPU Queuing or Machine Level Gate Serialization wait time appears substantially or overwhelmingly against Dispatched CPU Time while running important workload and reducing number of concurrent jobs is not possible.

Proportion between Dispatched CPU Time VS sum of all wait times is key to wait time analysis interpretation



To reduce high CPU Queuing or Machine Level Gate Serialization wait time during batch process period, consider reducing number of concurrent jobs first and observe run-time result. This can improve overall run-time. If this is not the case, add more CPU core(s).

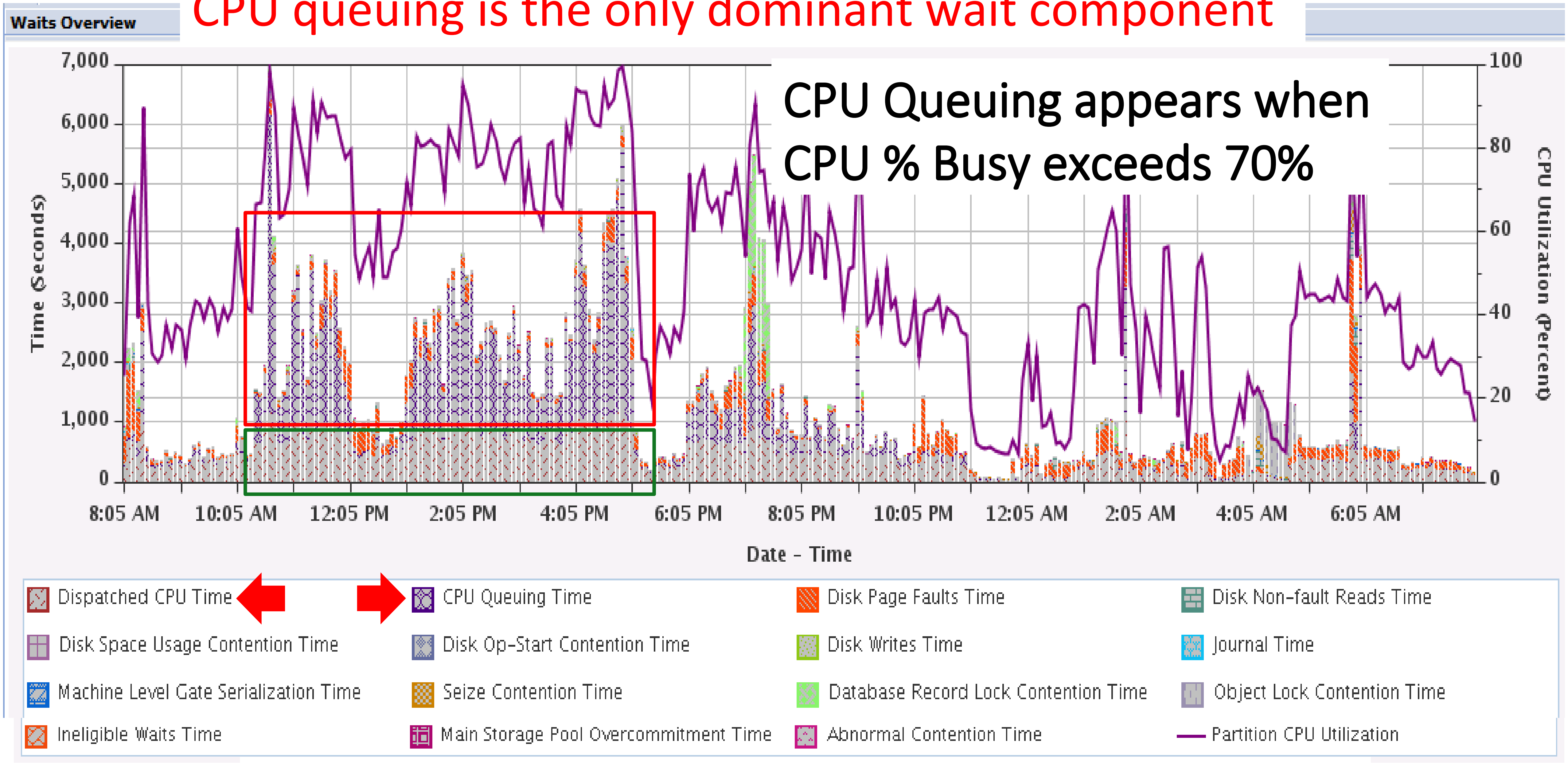
Rule of thumb: 6 concurrent jobs per CPU core (POWER8, 9, 10).

Persistent CPU % Busy at 90% or more but without or little CPU Queuing or Machine Level Gate Serialization wait time means there is no immediate lack of CPU power. But there remains system capacity sizing issue to be considered.

High CPU % Busy is not a reliable deciding factor on whether to add more core for better workload performance (as opposed to system capacity sizing) because as of POWER5-based server when simultaneous multithreading (SMT-2) was introduced up to POWER10 with SMT-8, POWER CPU can be highly busy without any CPU Queuing.

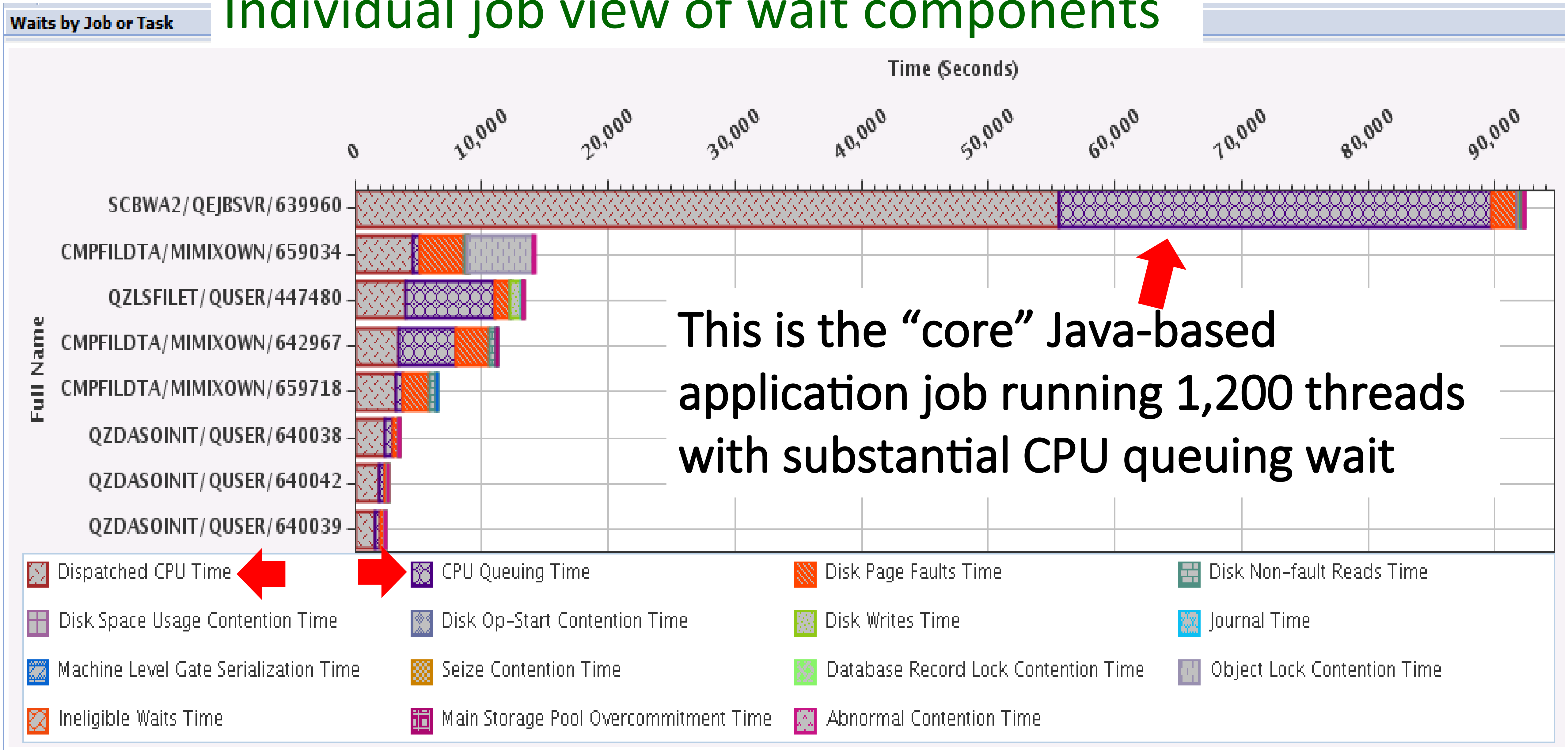
Let's look at sample analyses next.

CPU queuing is the only dominant wait component

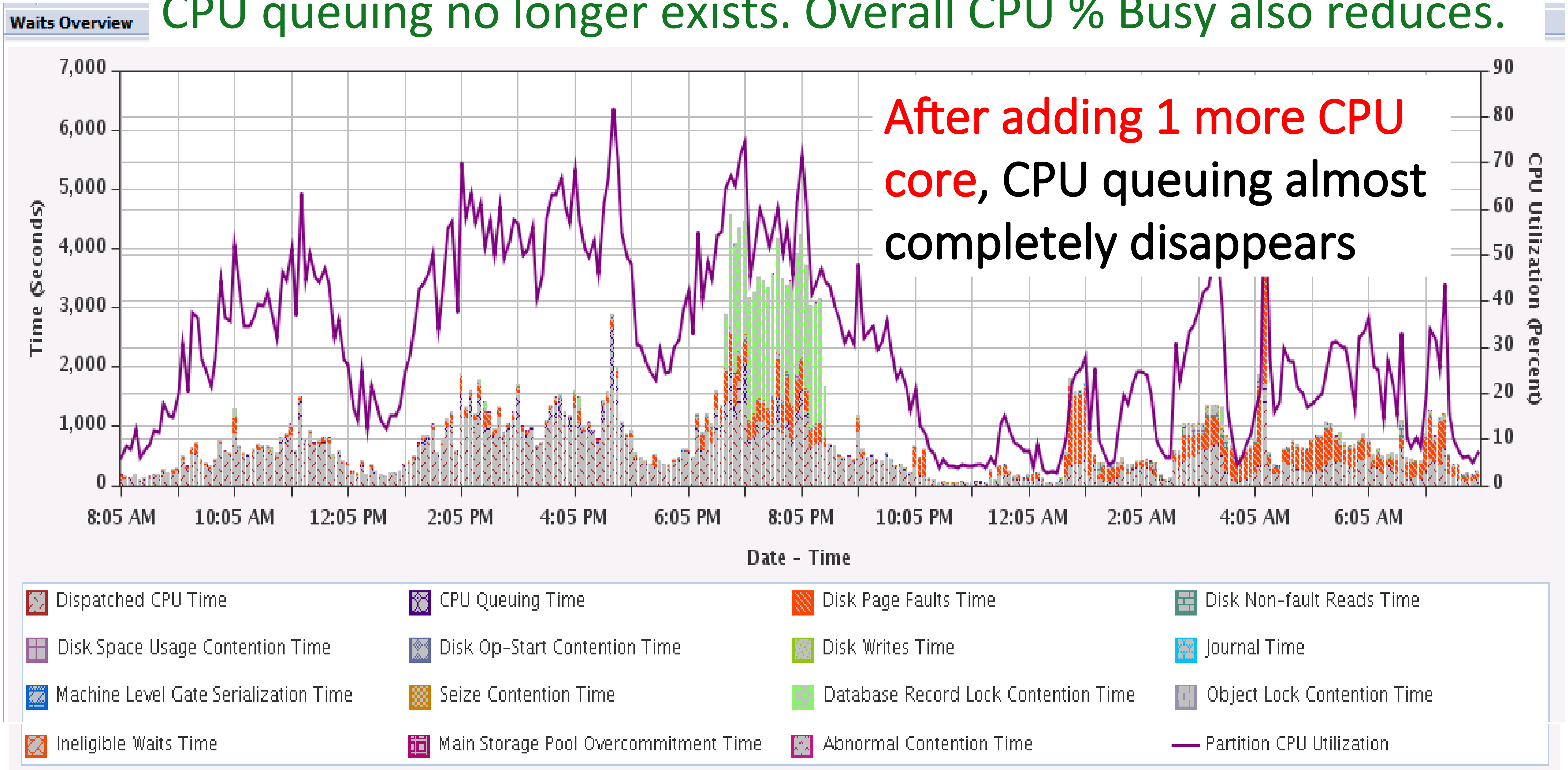


1 POWER8 core runs this Java-based workload.

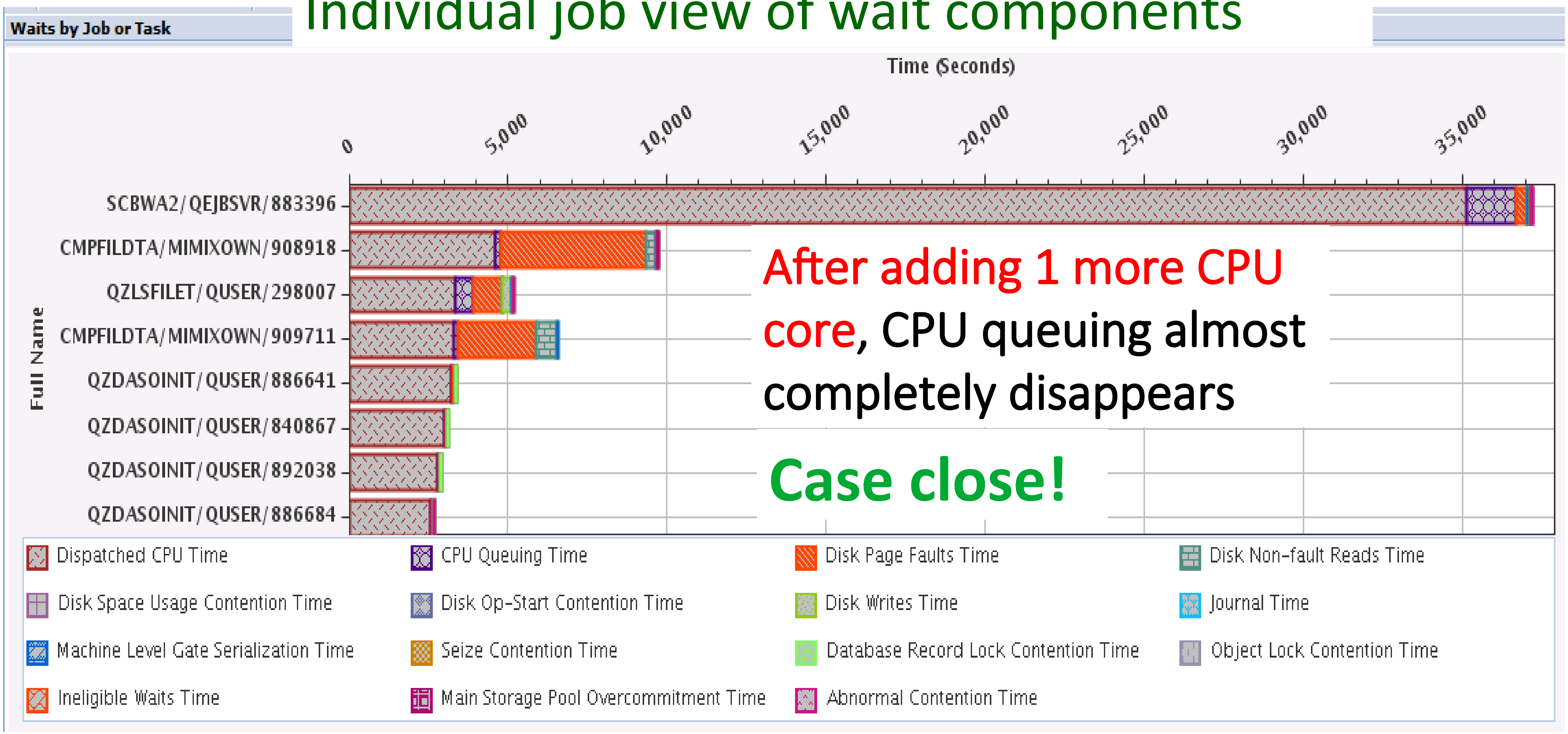
Individual job view of wait components



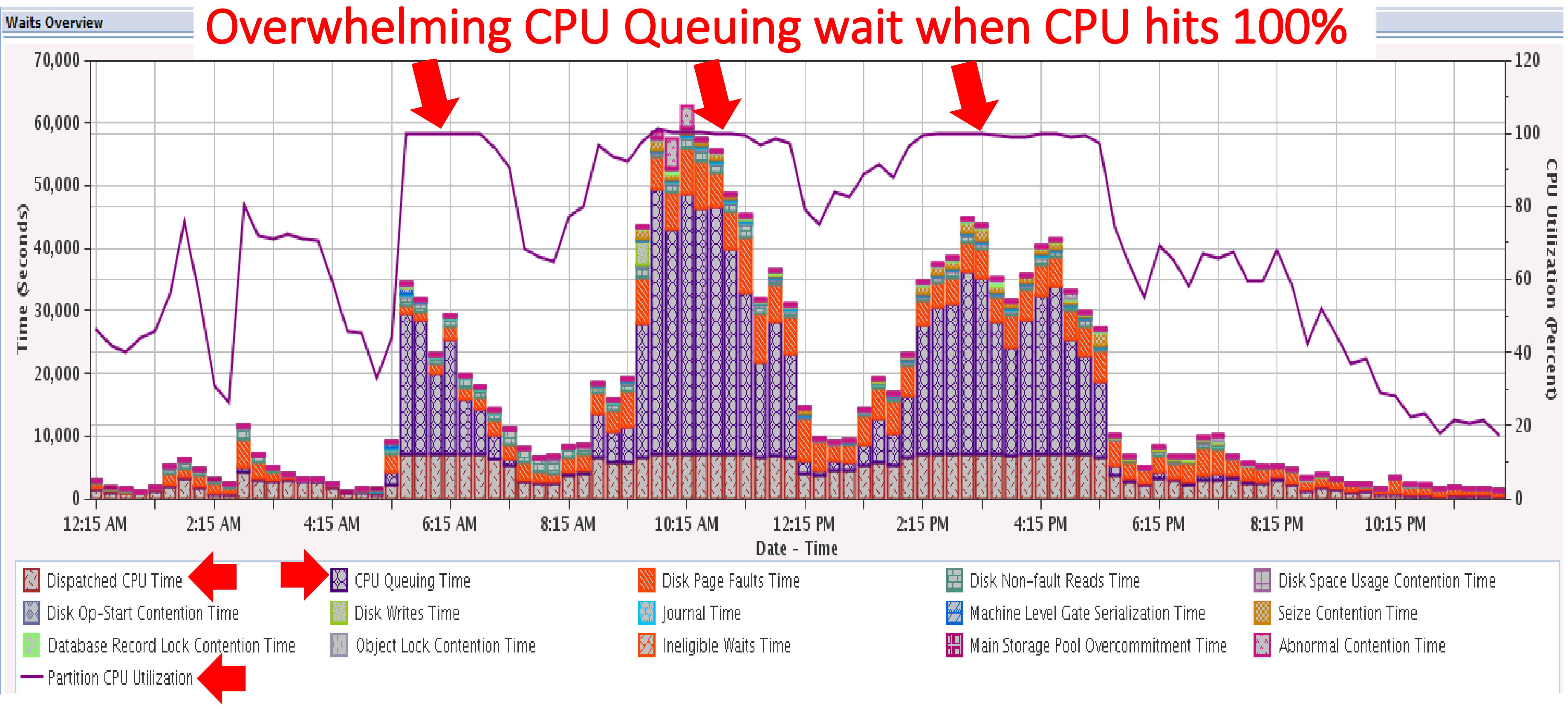
CPU queuing no longer exists. Overall CPU % Busy also reduces.



Individual job view of wait components

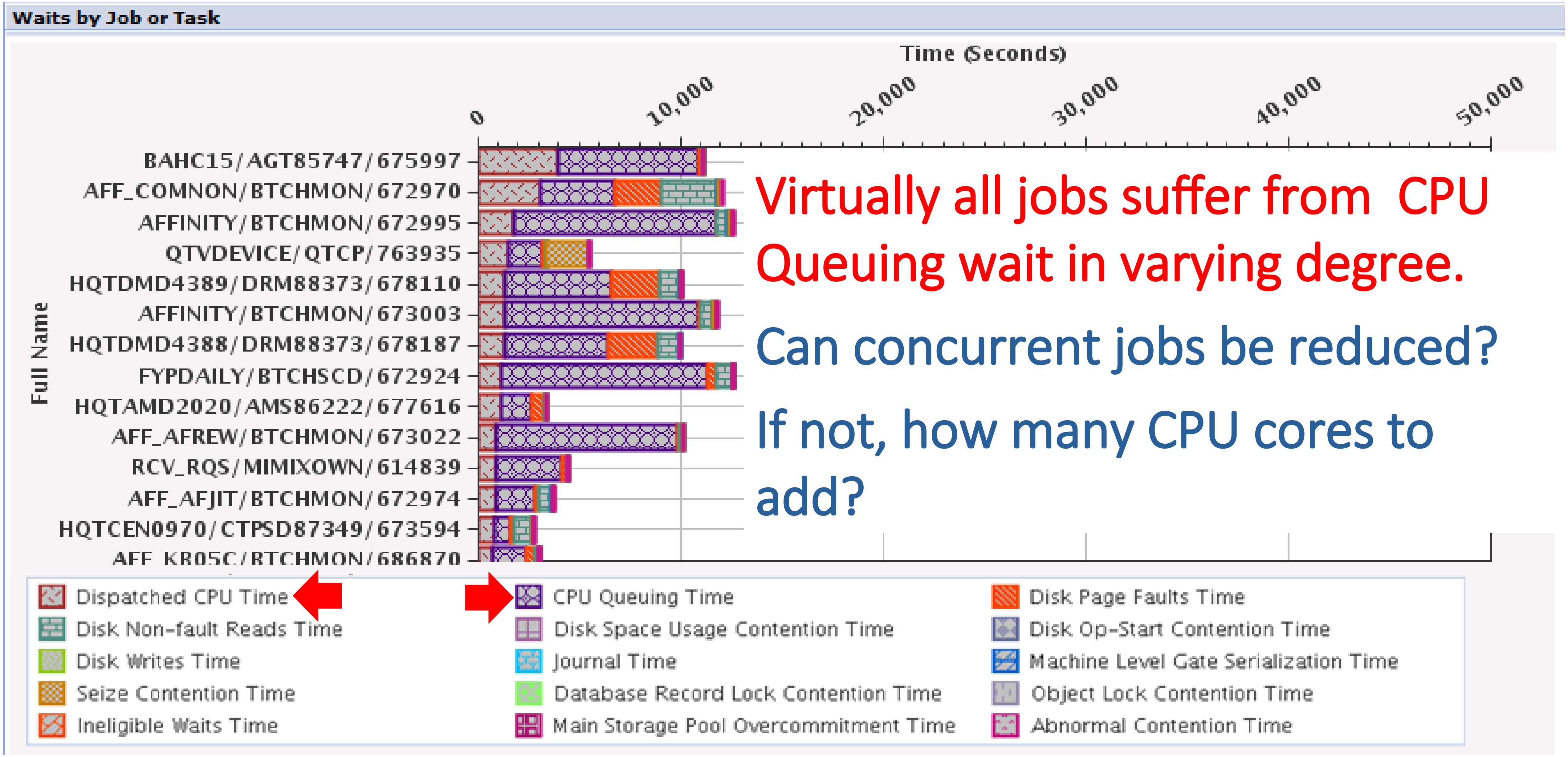


Another example – non-Java workload



5 CPU cores run this workload. Will adding 1 more core help?

Another example – non-Java workload



Virtually all jobs suffer from CPU Queuing wait in varying degree.

Can concurrent jobs be reduced?

If not, how many CPU cores to add?

If reducing concurrent jobs is not a viable solution, the question is how many more CPU cores are needed over the base 5 cores?

For enterprise class Power server, use Trial Capacity on Demand to find the answer. Trial CoD is free of charge for 30 days.

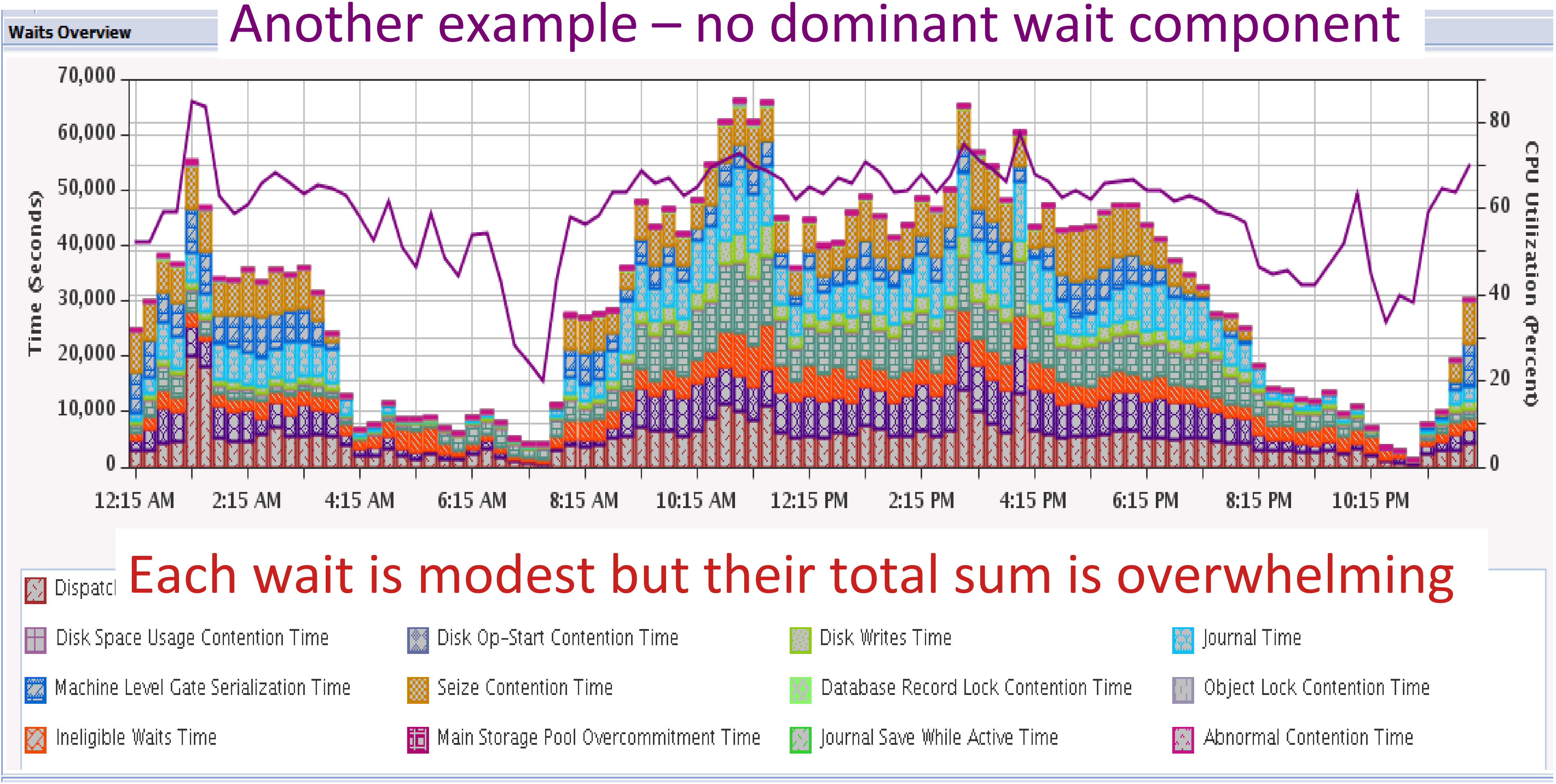
https://www.ibm.com/docs/en/power9/9223-42H?topic=demand-trial-capacity-concepts

For non-enterprise class server, buy Temporary IBM i License. This is charged per month.

https://www.ibm.com/common/ssi/ShowDoc.wss?docURL=/common/ssi/rep_ca/5/897/ENUS216-425/index.html

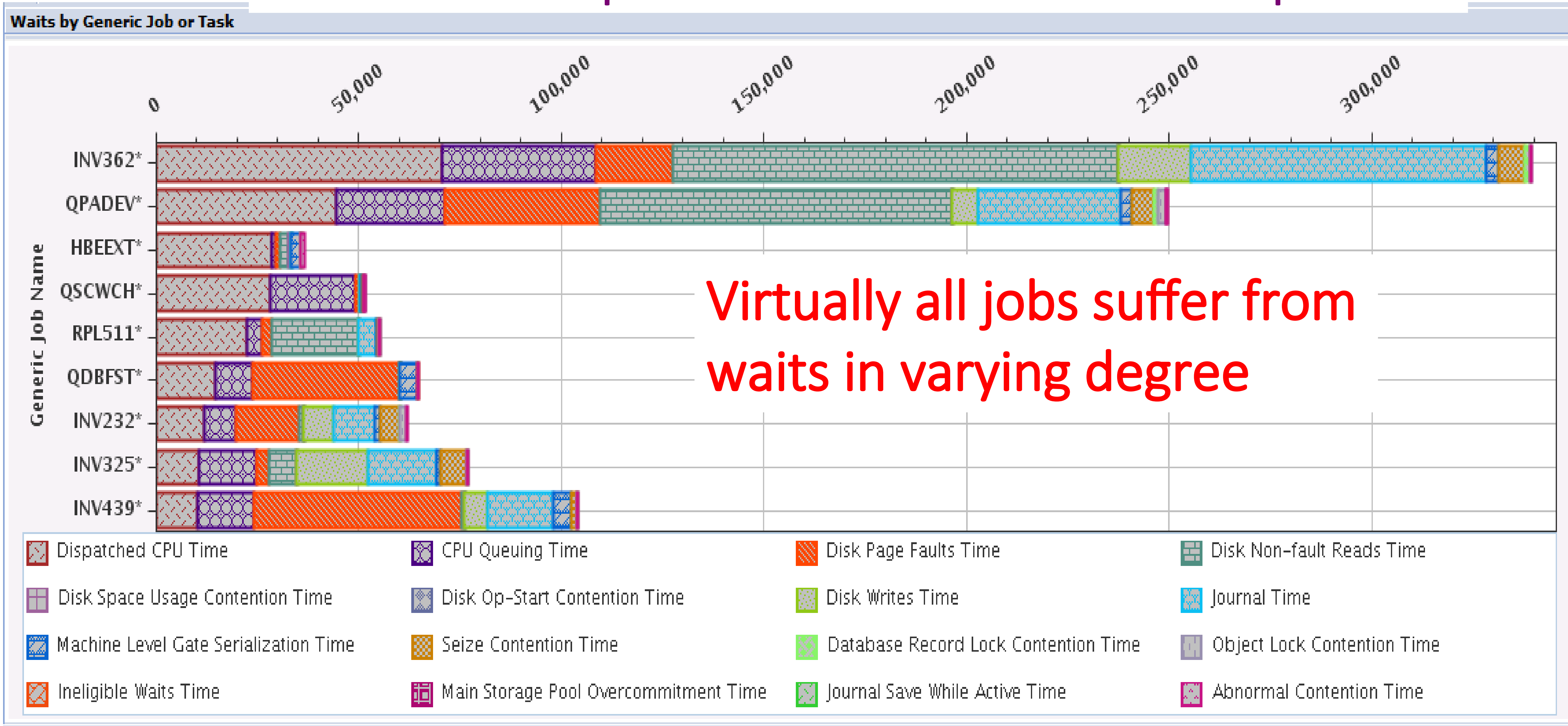
If many LPARs run in the same server, check if Uncapped Partitioning is used or not? You also need to use Shared Processor Pool for this to work.

Another example – no dominant wait component



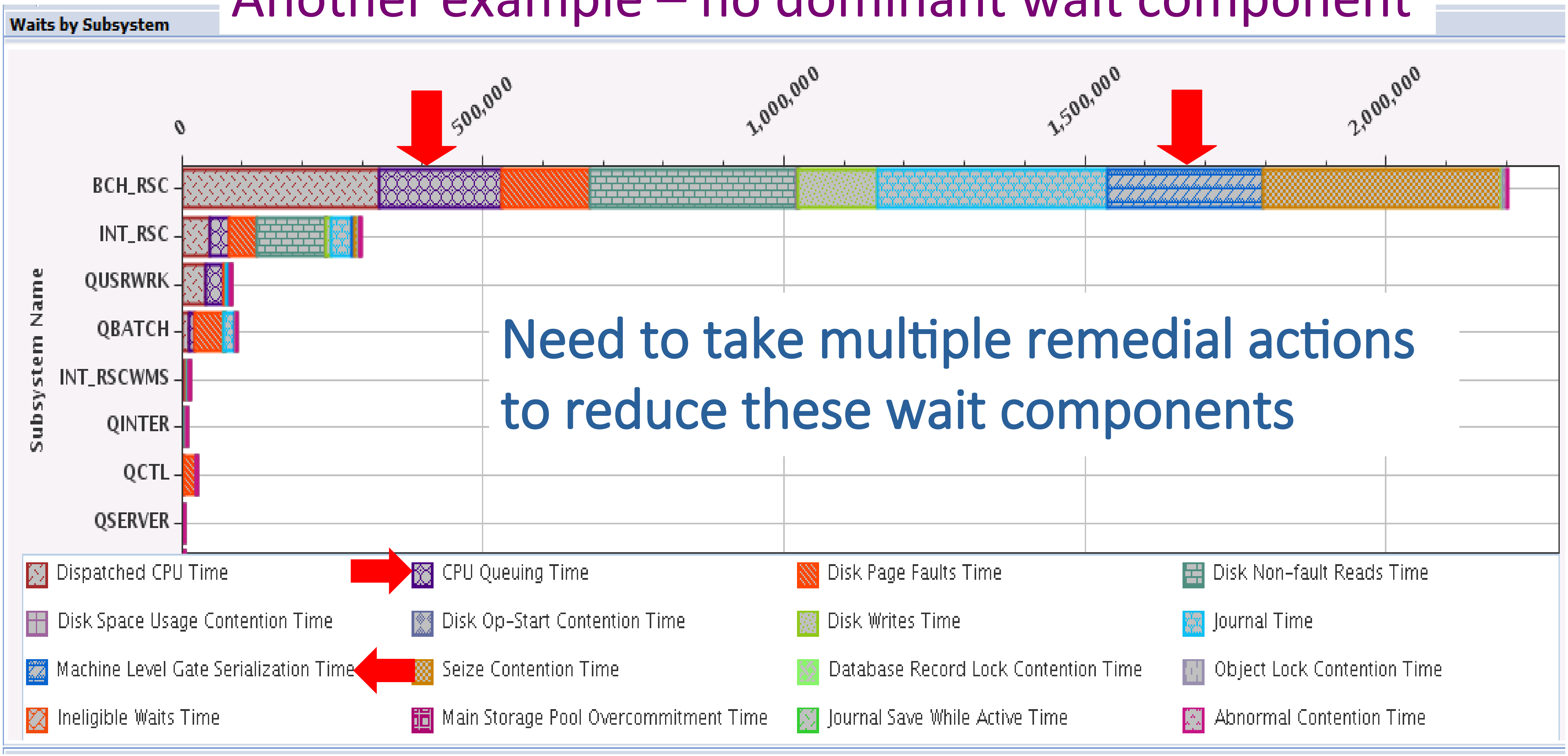
Each wait is modest but their total sum is overwhelming

Another example – no dominant wait component



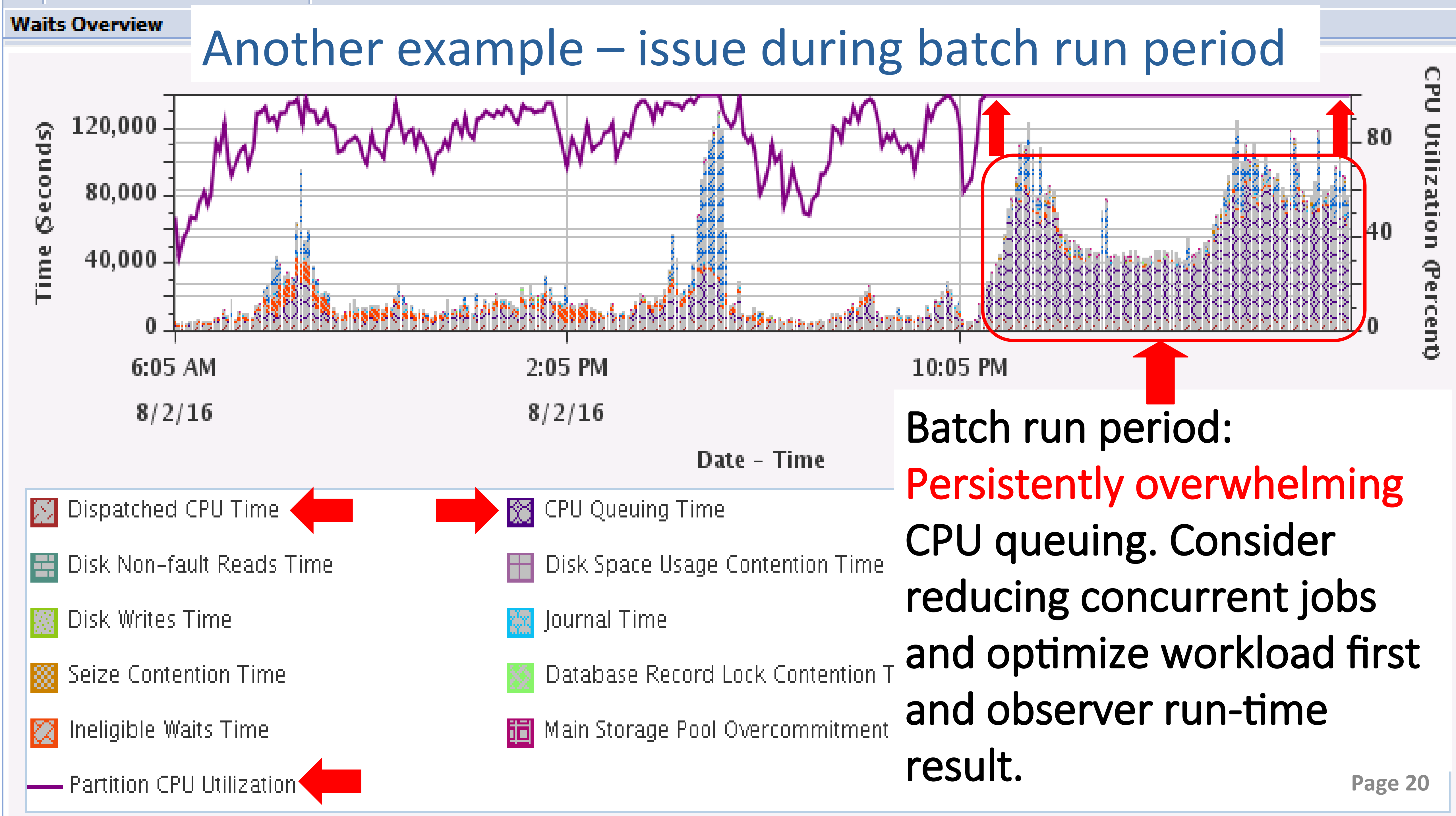
Virtually all jobs suffer from waits in varying degree

Another example – no dominant wait component



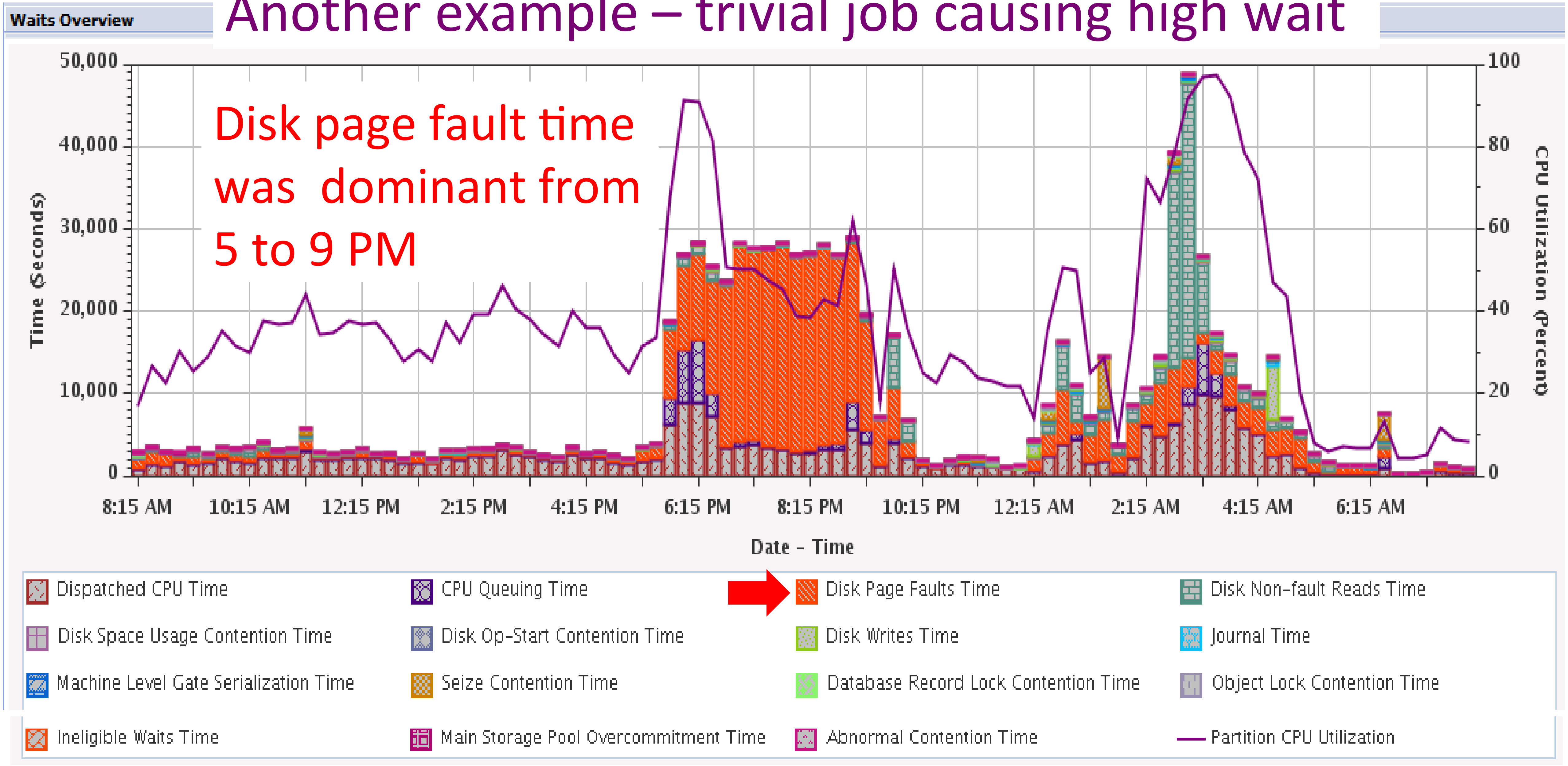
Need to take multiple remedial actions to reduce these wait components

Another example – issue during batch run period

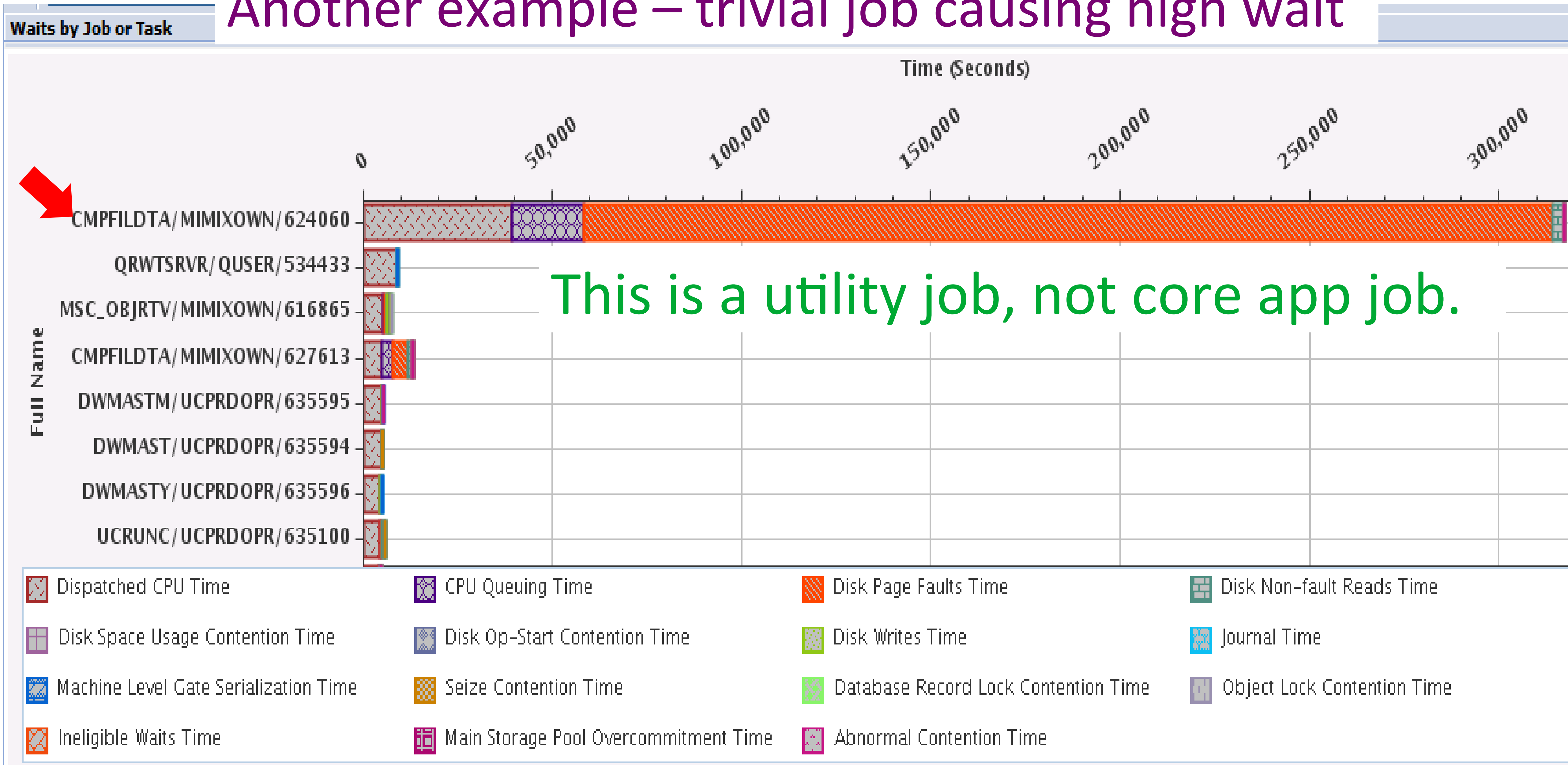


Batch run period: **Persistently overwhelming** CPU queuing. Consider reducing concurrent jobs and optimize workload first and observer run-time result.

Another example – trivial job causing high wait



Another example – trivial job causing high wait



Do we need to add more memory?

Use PDI chart on Memory Available by Pool.

PDI Memory by Pool charts were enhancement delivered via PTF for IBM i 7.3 and 7.4 in early 2020.

Look at the chart on several high/peak workload days before making a decision on which pool has persistent excess memory and which has persistently little or none left. This helps you move memory among pools for optimal use.

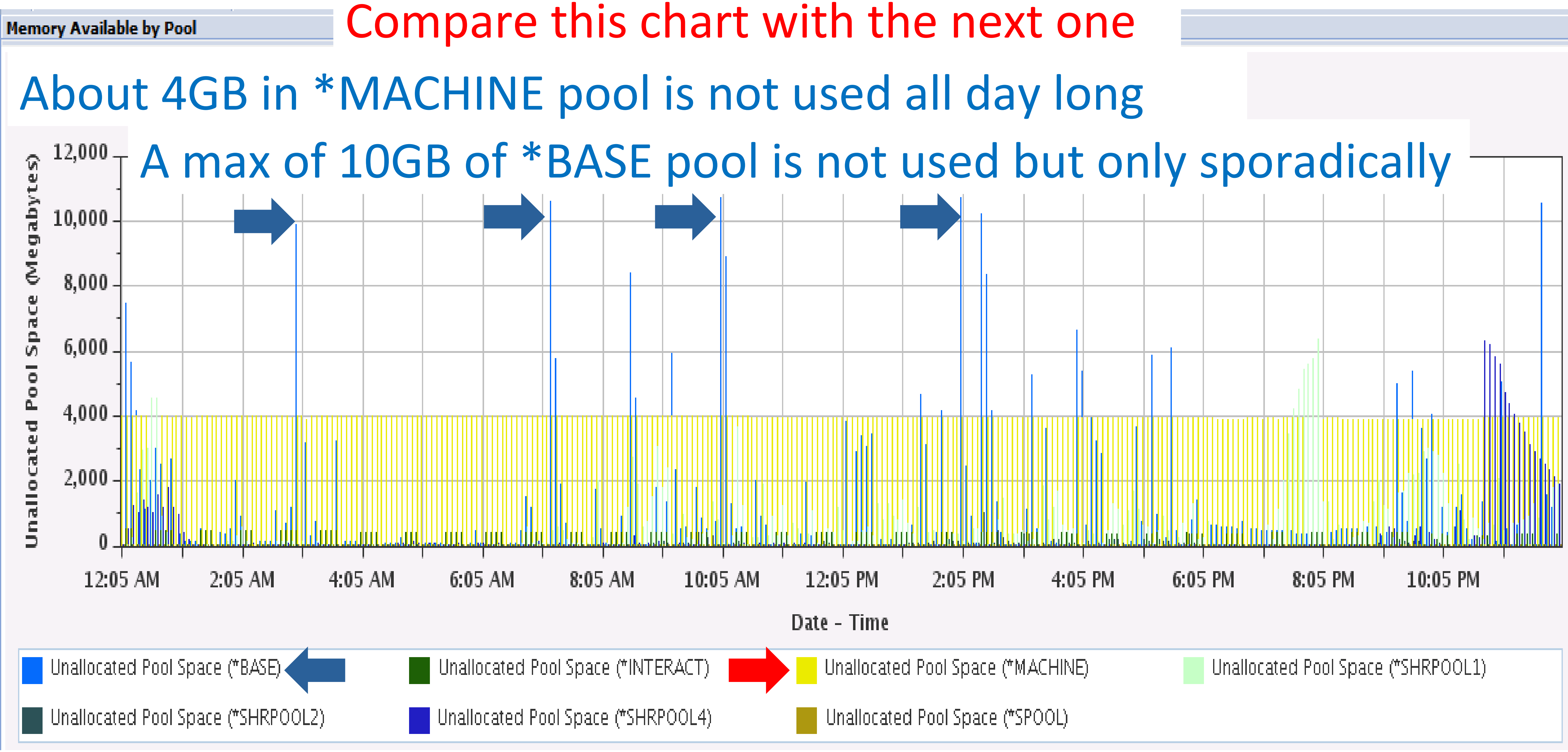
Learn to use WRKSHRPOOL command to put lower and upper limits to each pool after reviewing the charts.

Let's look at a sample analysis.

Compare this chart with the next one

About 4GB in *MACHINE pool is not used all day long

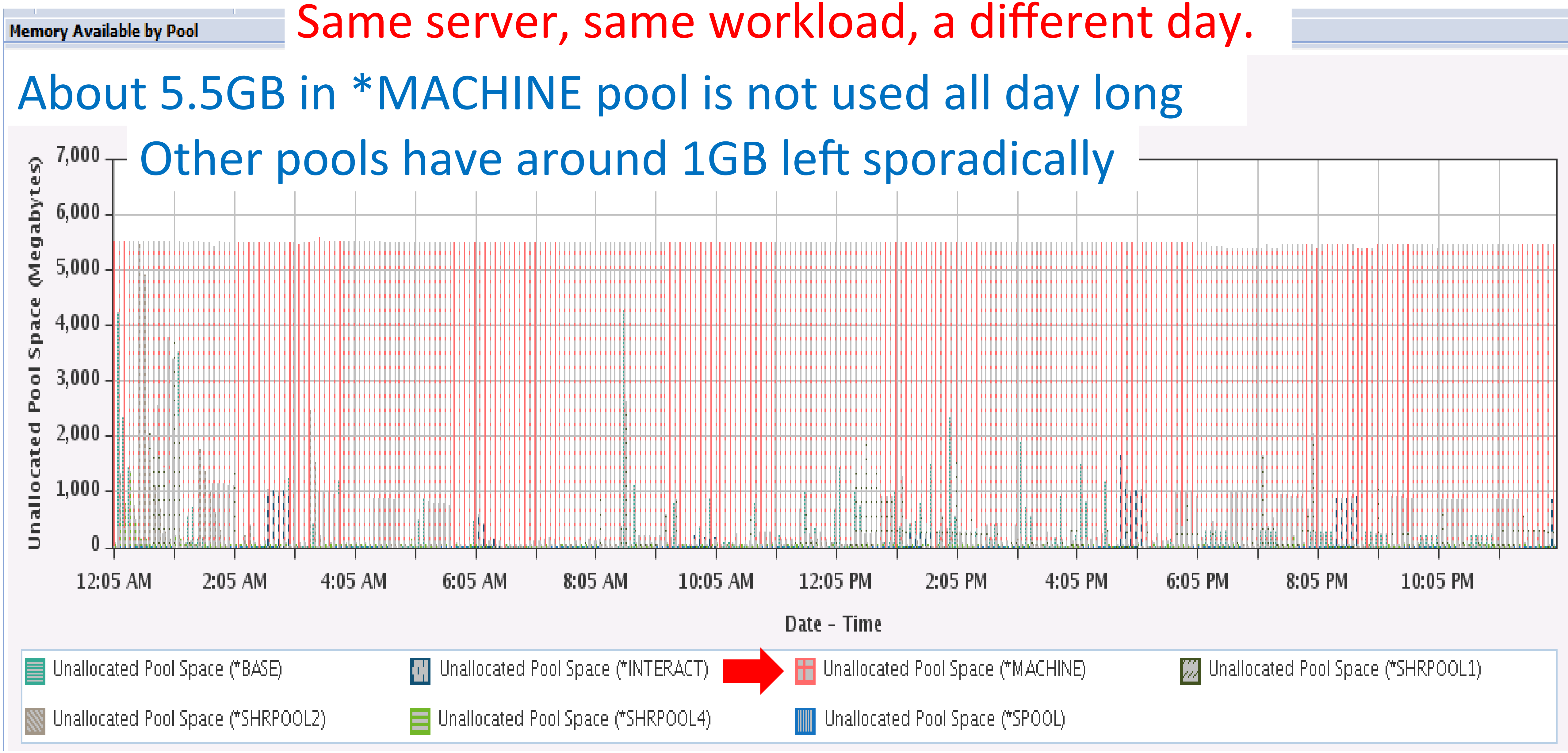
A max of 10GB of *BASE pool is not used but only sporadically



Same server, same workload, a different day.

About 5.5GB in *MACHINE pool is not used all day long

Other pools have around 1GB left sporadically



From the charts, *MACHINE pool is the only pool with persistent excess memory left all day long. Its size should be reduced and have its maximum fixed by WRKSHRPOOL command.

Distribute the excess memory to *INTERACT, *SHRPOOL1, 2, and 4.

Produce the charts again and repeat the process of resizing the pools until high amount of excess memory is no longer seen.

Do we need to add more memory to our server?

Use "evidence of absence" in the chart Memory Available by Pool. If you see "empty" charts on several high/peak workload days, it's time to add more memory to the server because you see no excess memory at all.

I have multiple disk pools (ASPs). How does each perform?

Use PDI charts on Disk Overview for Disk Pools and Disk Overview by Disk Unit.

Rule of thumb: Good disk response time guideline is 5 millisecond or less for HDD, 2.5 millisecond or less for SSD/Flash disk.

Let's look at some sample analyses.

Installed Disk Hardware

--- Select Action ---

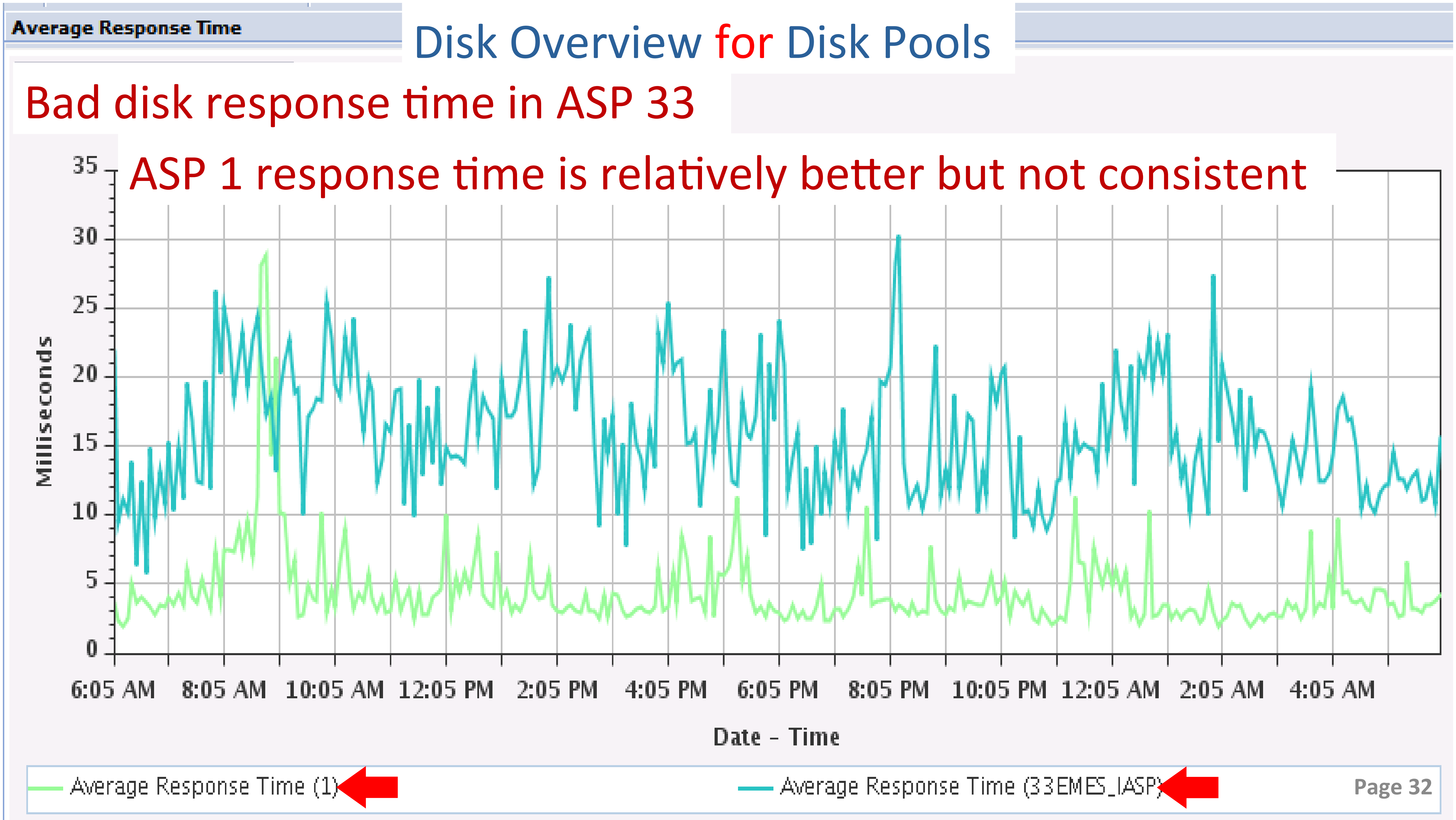
Select	ASP Number	Disk Unit Type	Feature Code	RAID Type	Unit Count	ASP Capacity (GB)	Disk Used	Average Unit Size
☐	1	EMC		RAID-5	192	13824.4	12.14	72
☐	33	EMC		RAID-5	640	46081.4	69.28	72

Total: 2 Filtered: 2

Disk Overview for Disk Pools

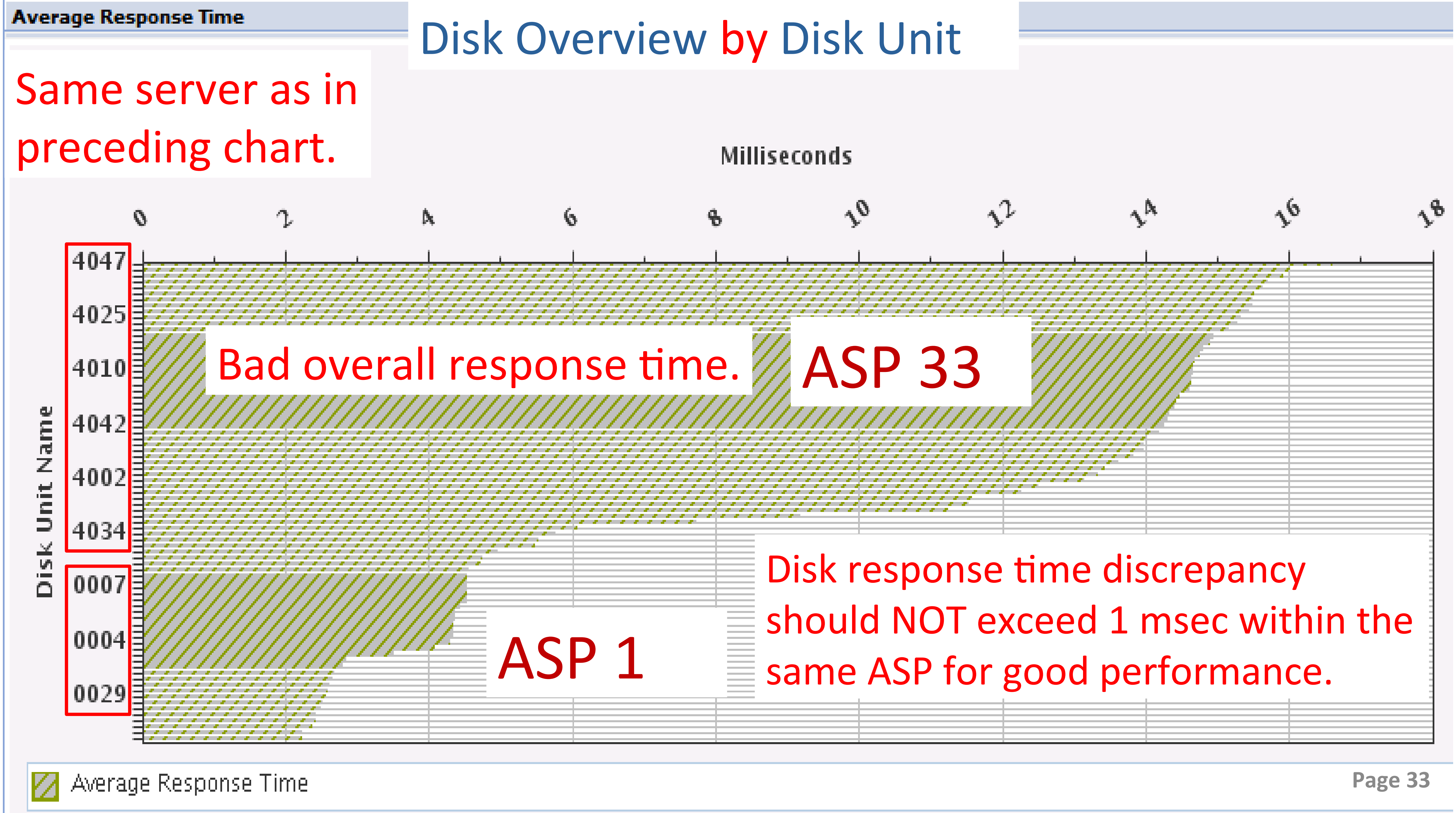
Bad disk response time in ASP 33

ASP 1 response time is relatively better but not consistent



Disk Overview by Disk Unit

Same server as in preceding chart.



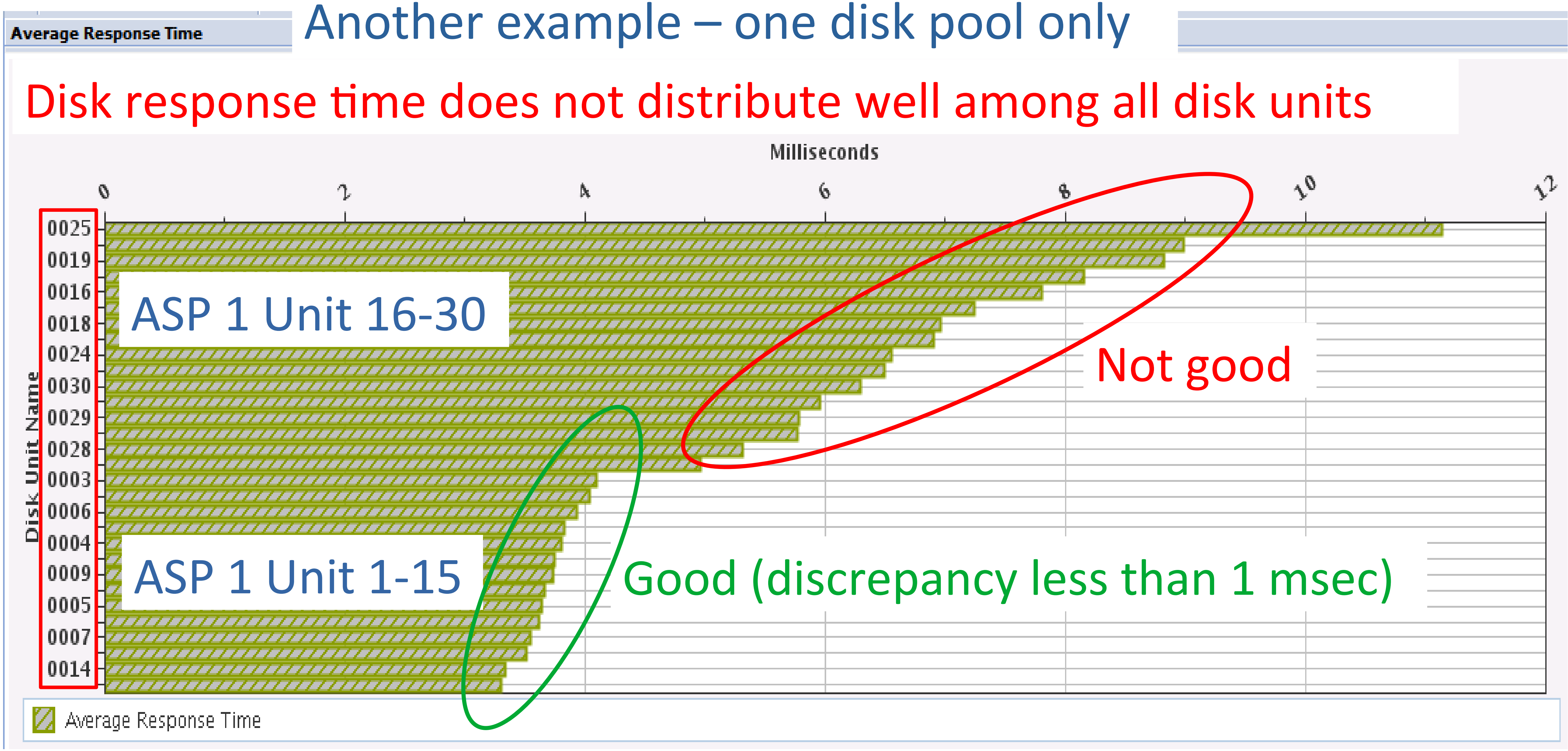
Bad overall response time.

ASP 33

ASP 1

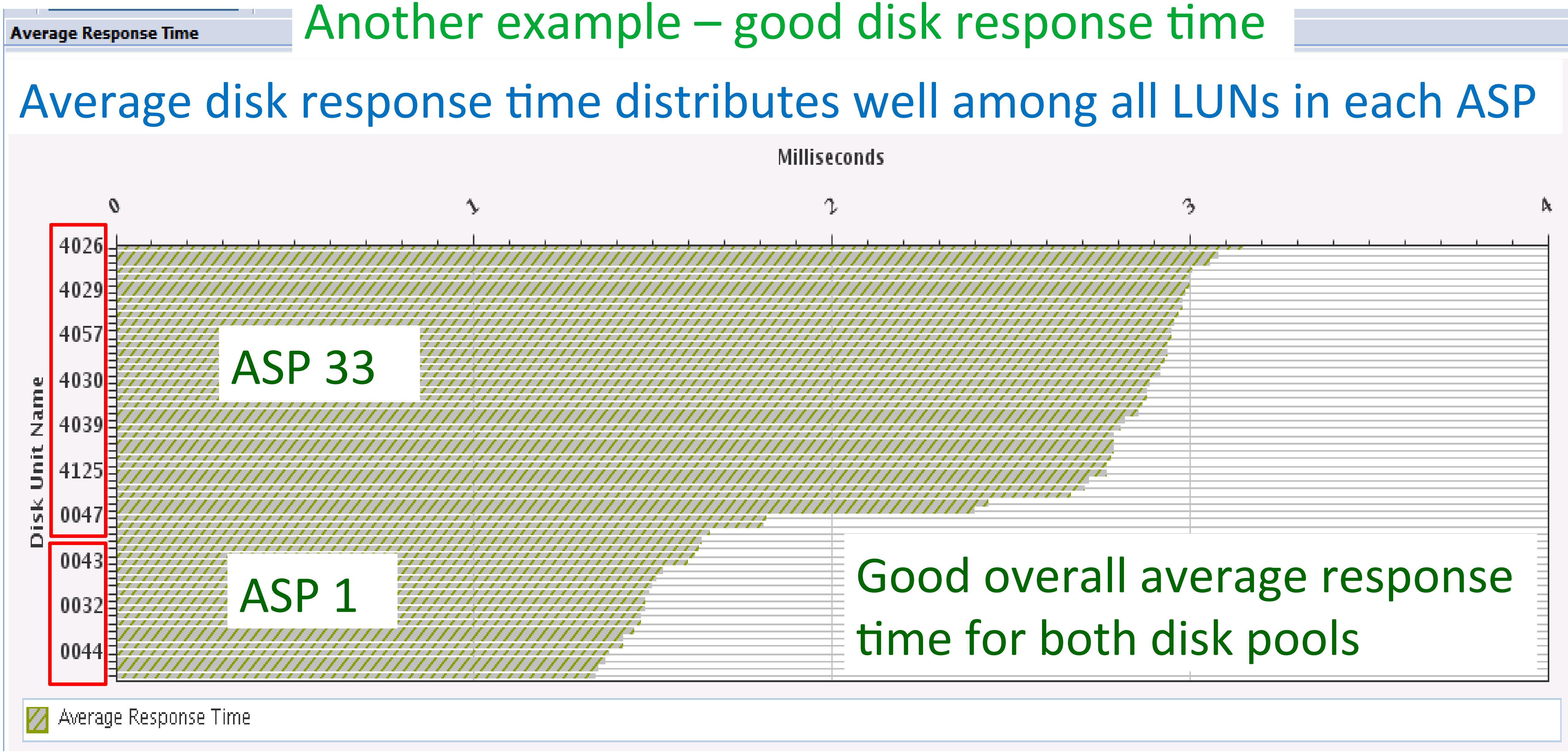
Disk response time discrepancy should NOT exceed 1 msec within the same ASP for good performance.

Another example – one disk pool only



Another example – good disk response time

Average disk response time distributes well among all LUNs in each ASP



Do we have workload growth (or reduction)?

Use PDI charts on Resource Utilization Rates and look at Total Logical Database I/Os Per Second component.

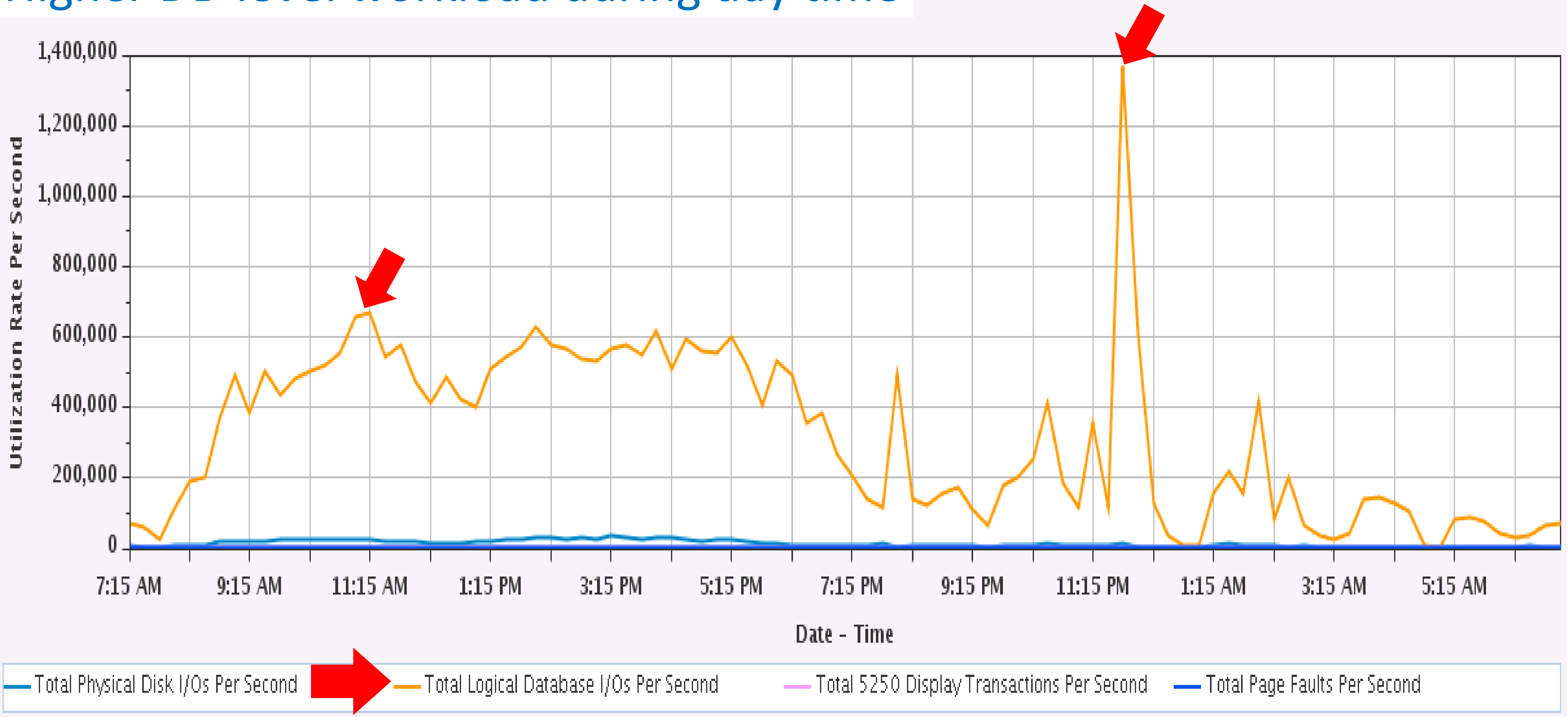
Why not use CPU % Busy as an indicator? This is not consistently reliable in many cases, For example, application “tuning” action(s) can reduce CPU % busy while Logical DB IOPS may even increase.

Look at multiple charts from multiple high/peak workload days or servers to make meaningful comparison.

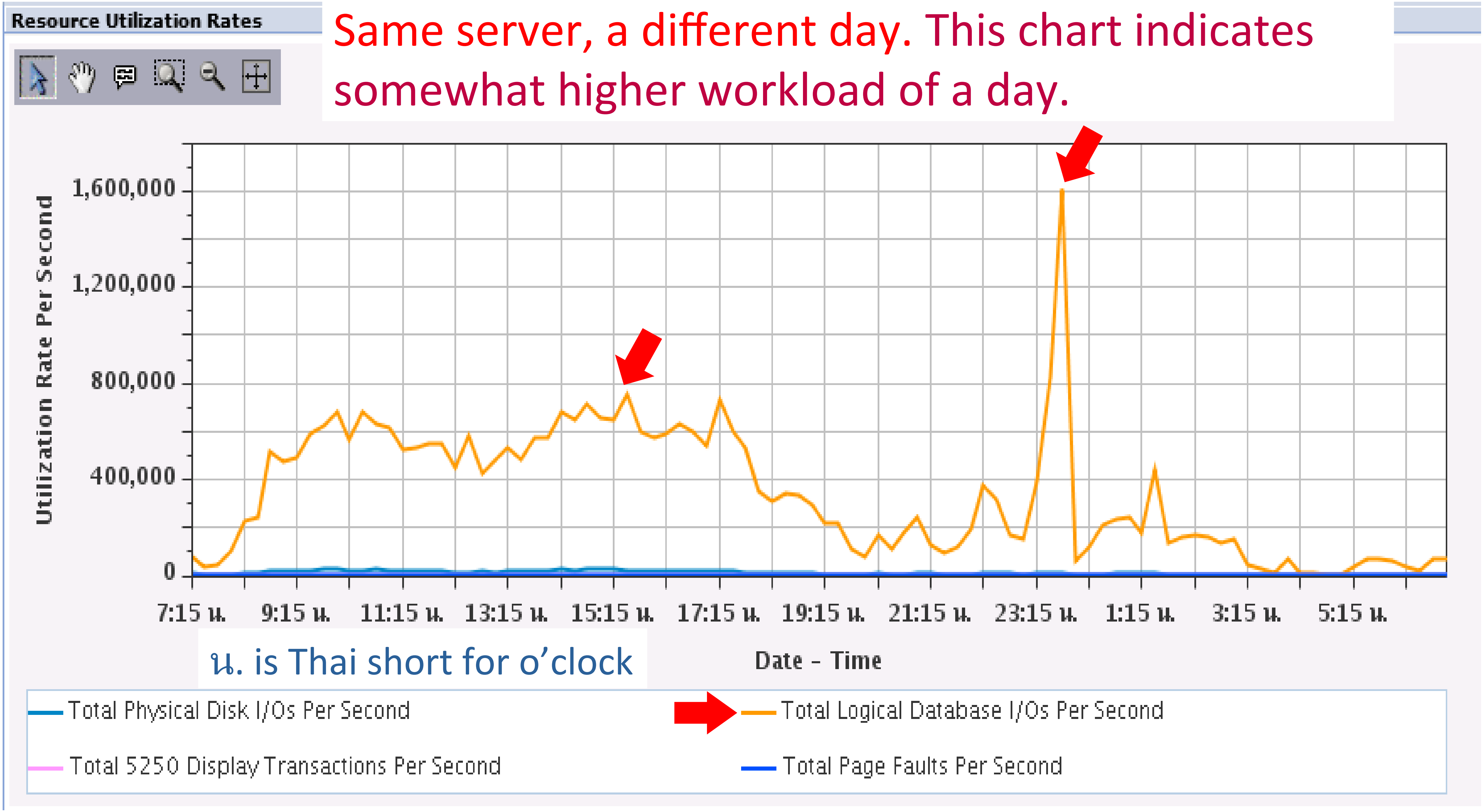
Resource Utilization Rates

Compare this chart to the next one

Higher DB-level workload during day time

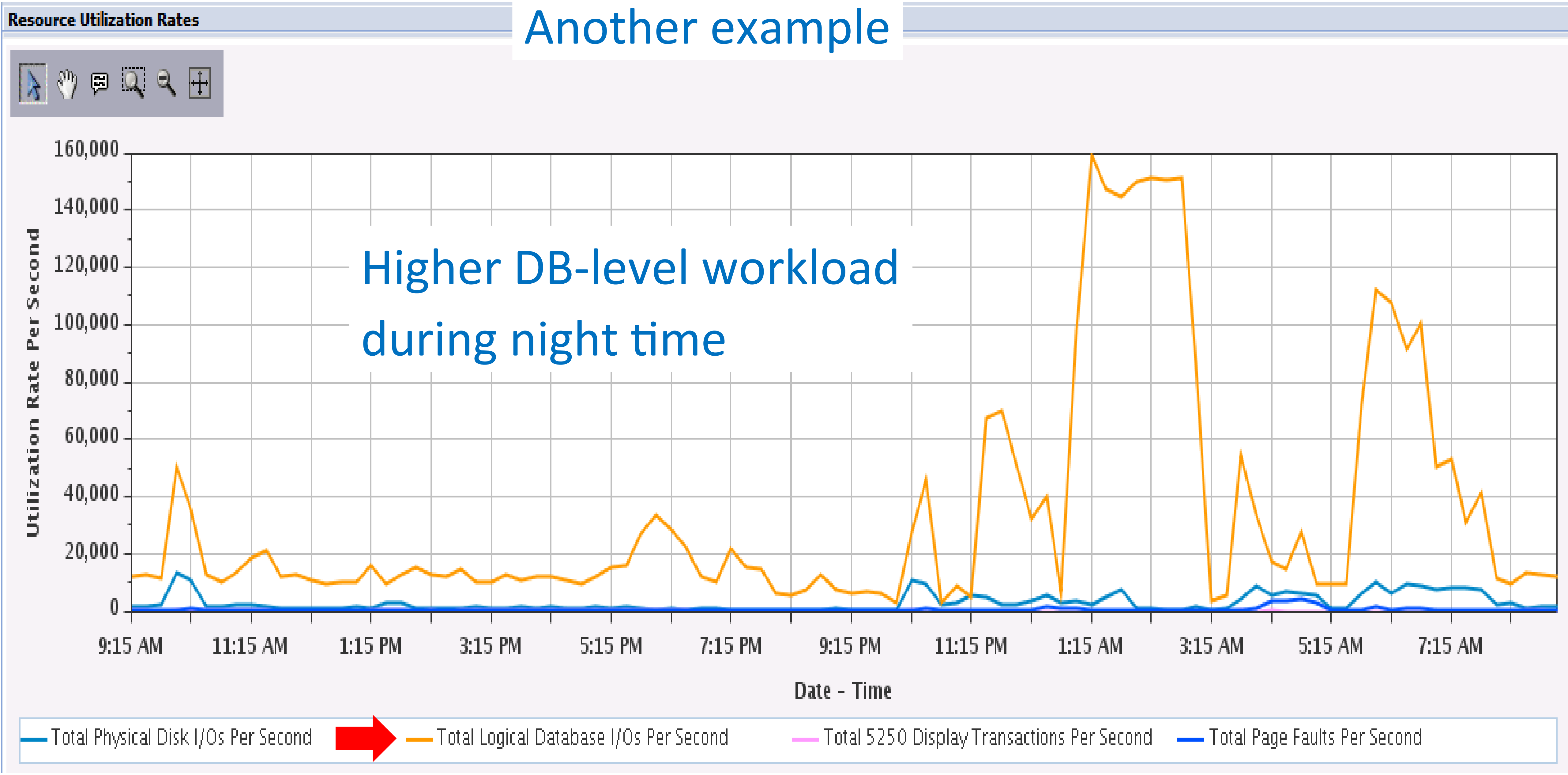


Same server, a different day. This chart indicates somewhat higher workload of a day.



น. is Thai short for o'clock

Another example



Does performance tuning works?

Start with PDI charts on Wait Overview and Wait by (Generic) Job or Task and Wait by Subsystem.

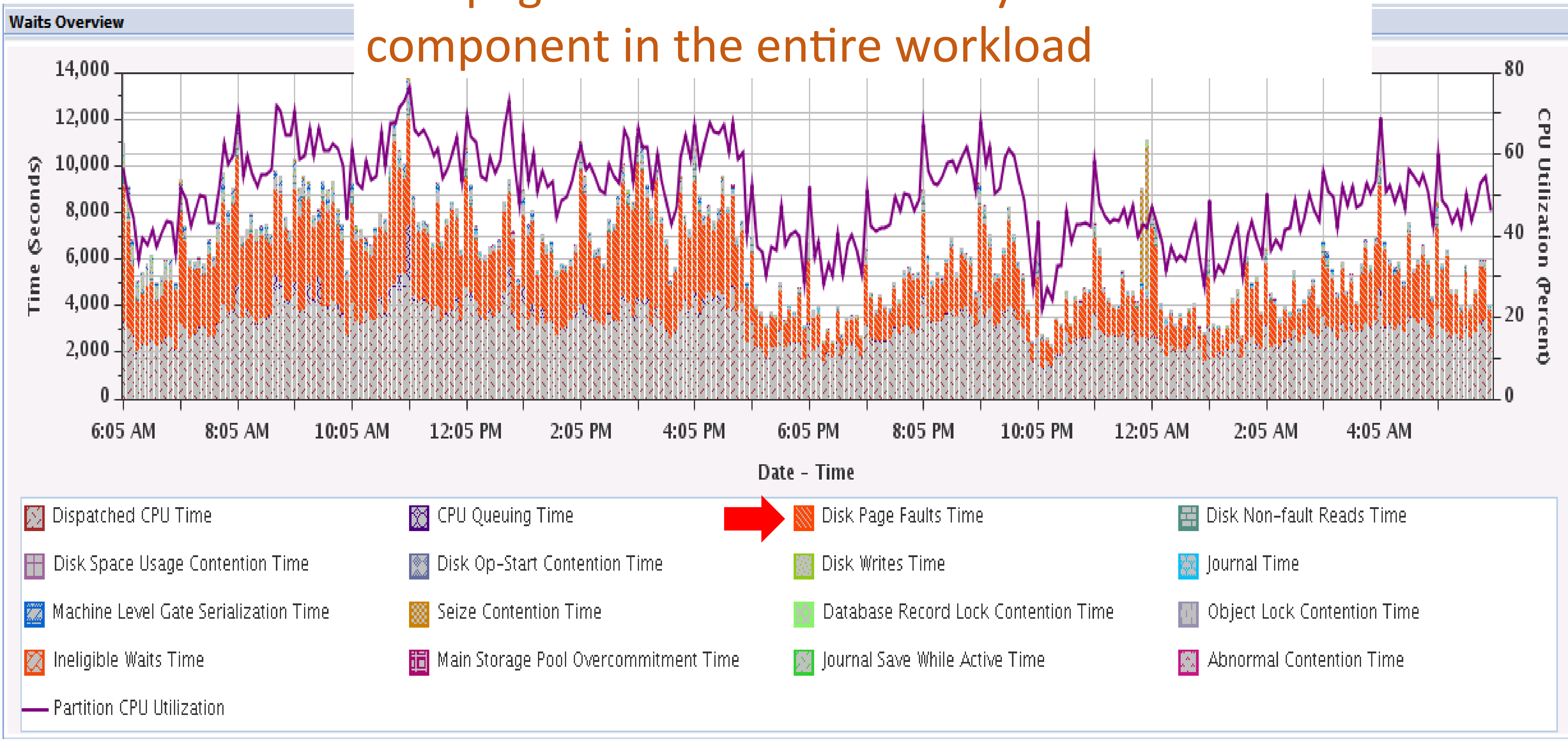
Identify dominant wait component(s) with substantial to overwhelming ration against Dispatched CPU Time.

Identify the cause of the dominant wait and how to address it. Then take proper action(s) to attack dominant wait component(s).

Display wait charts again to check for the improvement.

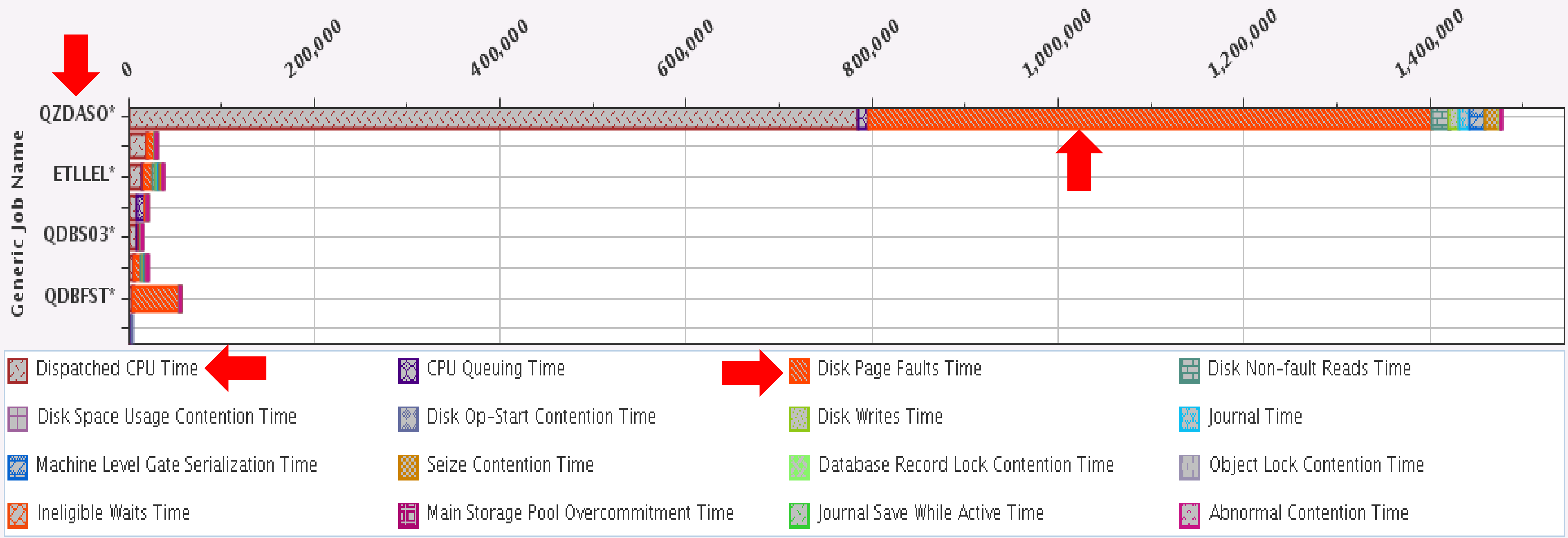
Let's look at a sample analysis.

Disk page fault time is the only dominant wait component in the entire workload



Waits by Generic Job or Task

DB2i remote SQL jobs (QZDASOINIT) carry almost all disk page fault wait component



QZDASOINIT is DB2 for i job serving remote SQL from ODBC/JDBC.

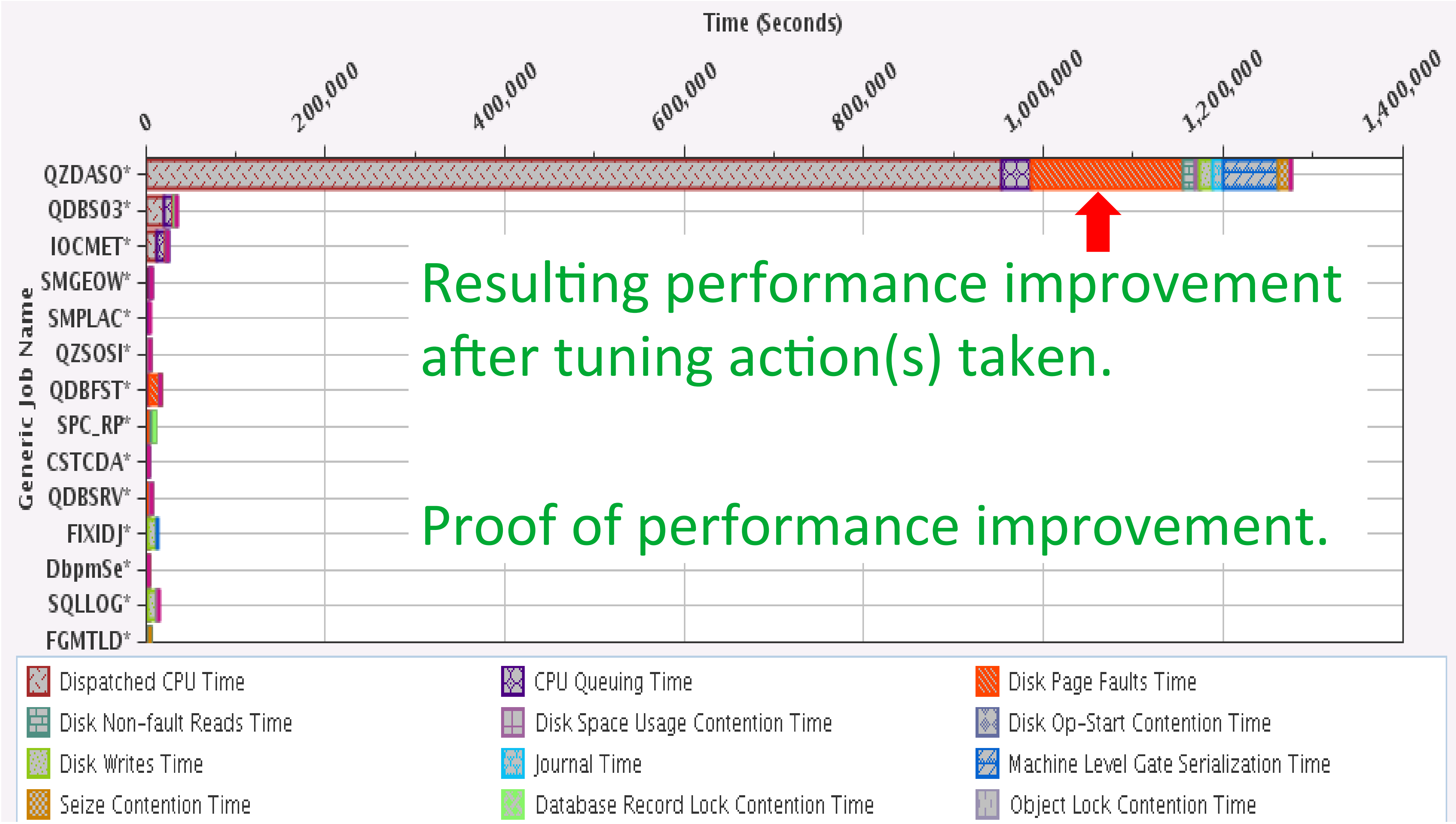
This customer runs Java-based core business application in many Intel servers that submit SQL via JDBC to DB2 for i. This is why all QZDASOINIT jobs consume almost all of CPU times in the server as seen in the previous chart.

Typically, SQL workload without sufficient number of useful indexes for optimal SQL workload performance causes excessive memory faulting rate which leads to Disk Page Faults Time wait as seen in the chart. Here, memory faulting is mainly caused by excessive table scans made by SQL engine.

Useful tools are available in DB2 for i for use to identify and create useful indexes to help reduce excessive memory faulting: Plan Cache Snapshot Analyser, Visual Explain, Index Advisor, and Index Condenser.

After useful indexes are created, produce Wait charts again to see the result. Look at the next chart.

Waits by Generic Job or Task



Resulting performance improvement after tuning action(s) taken.

Proof of performance improvement.

IBM i Performance Data Investigator (PDI) tool

Note: All charts in this presentation are from PDI tool of heritage version of Navigator for i that relies on the vulnerable Log4j. Readers are encouraged to move to the new Navigator for i as soon as they can.

https://www.ibm.com/support/pages/node/6483299

https://www.youtube.com/watch?v=iVgrD8CMj9Q

Thank You