

# 9 Workload Management

*Are database systems just another application running in the virtualized environment?*

***No!*** *Virtualization poses several interesting research questions for database systems*

- Understanding the performance of database systems on virtual machines
- Configuring and tuning virtual machines running and tuning virtual machines running database systems
- Taking advantage of virtualization capabilities in the database system

*Overhead*

- TPC-H / PostgreSQL
  - Direct machine
  - Xen virtual machine
- Overhead of virtualization is not unacceptably high
- Can be made lower with better virtualization support

| | Base Runtime (secs) | Xen Runtime (secs) | Abs SlwDwn (secs) | Rel SlwDwn (%) |
|---|---|---|---|---|
| Q1 | 14.19 | 15.30 | 1.11 | 7.82 |
| Q3 | 5.20 | 6.98 | 1.78 | 34.35 |
| Q5 | 4.53 | 5.99 | 1.46 | 32.21 |
| Q7 | 4.09 | 5.32 | 1.23 | 30.14 |
| Q9 | 10.99 | 12.81 | 1.81 | 16.49 |
| Q10 | 5.04 | 6.36 | 1.32 | 26.17 |
| Q13 | 14.02 | 15.27 | 1.25 | 8.93 |
| Q18 | 9.38 | 11.54 | 2.17 | 23.12 |
| Q19 | 5.26 | 6.33 | 1.07 | 20.41 |
| Q21 | 2.79 | 3.65 | 0.86 | 31.03 |

*What level of resources to give to each DBMS?*

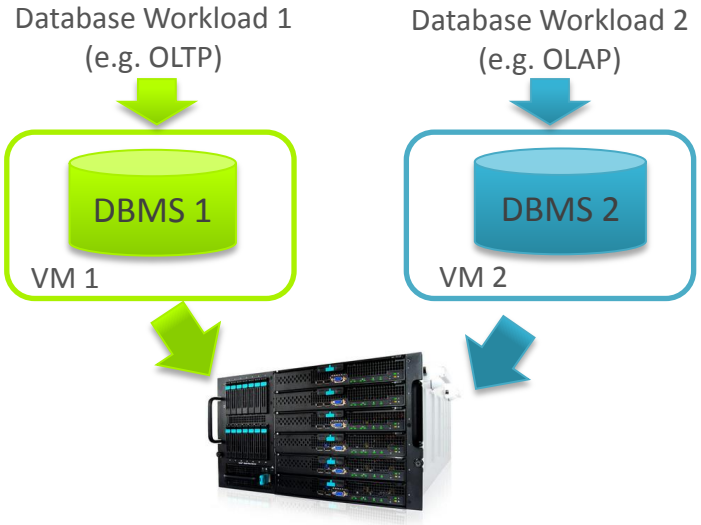
- Configuring VM parameters

*How to tune the DBMS for a given level of resources?*

- Configuring the DBMS parameters

*Need a **model** of how resource allocation affects database performance*

*Need optimization or control algorithms to decide on the optimal resource allocation*

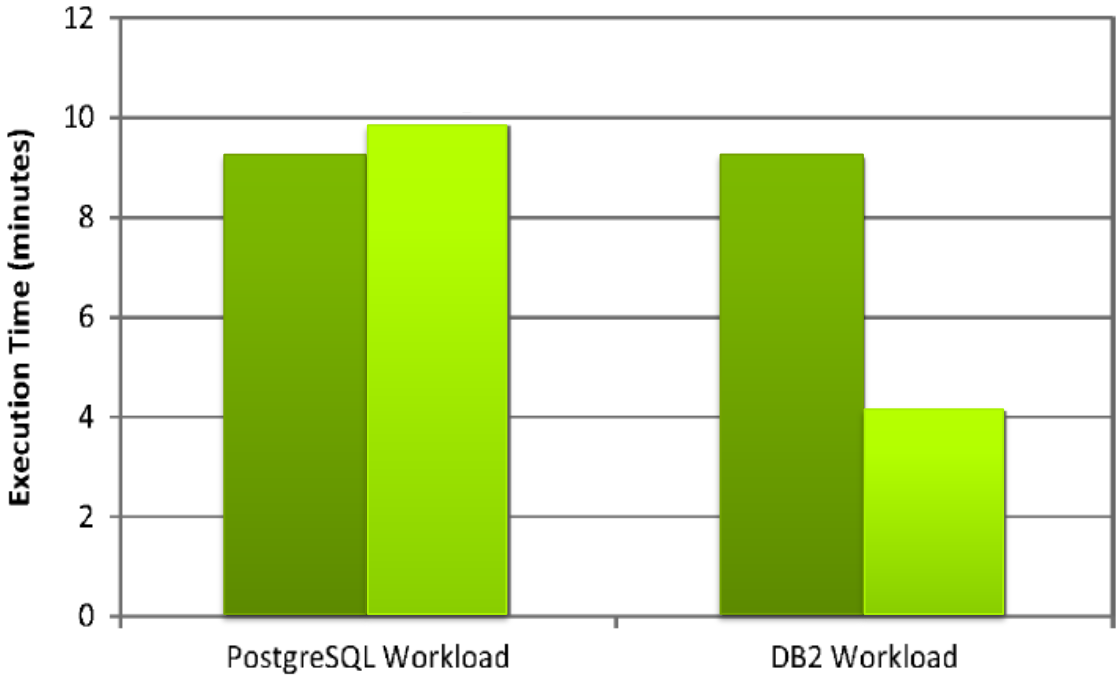


*Virtual Machine 1*

- PostgreSQL, 10 GB TPC-H, Query 17

*Virtual Machine 2*

- DB2, 10 GB TPC-H, Query 18




*Configuration 1*

- CPU: 50/50
- Memory: 50/50

*Configuration 2*

- CPU: 15/85
- Memory: 20/80

# *Virtualization Design Advisor*

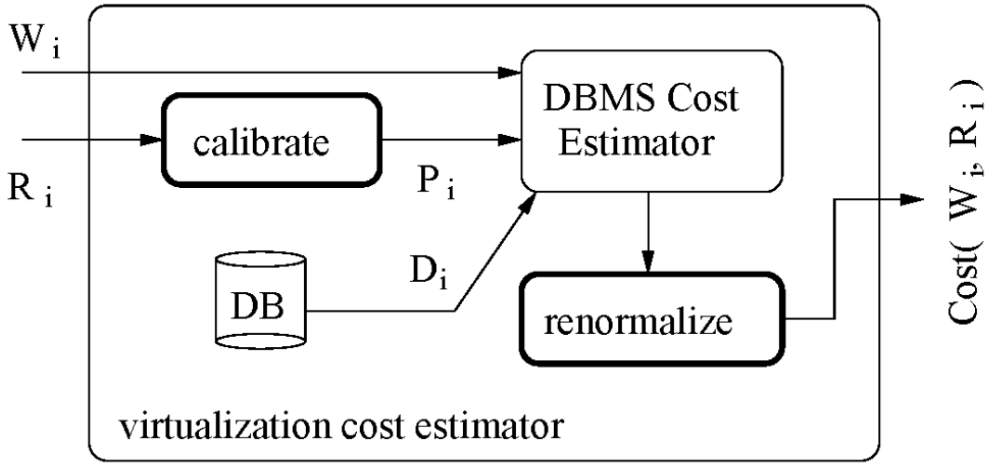
*Optimize performance of database management systems by controlling the configurations of the virtual machines in which they run*

- *Virtualization Design Advisor* recommends workload-specific configuration using the query optimizer's cost model (offline)
- Runtime information is collected for online refinement and to address changing workloads

*Problem Statement*

- $W_i$: set of workloads processed by i-th VM
- $R_i$: resource shares (here CPU + memory)
- $P_i$: query optimizer parameters
- $D_i$: database instance
- Optimization criteria: minimize

$$\sum_{i=1}^{N} Cost(W_i, R_i)$$




- Constrained by degradation limit

$$Degradation(W_i, R_i) = \frac{Cost(W_i, R_i)}{Cost(W_i, [1, \ldots, 1])}$$

$$Degradation(W_i, R_i) \leq L_i$$

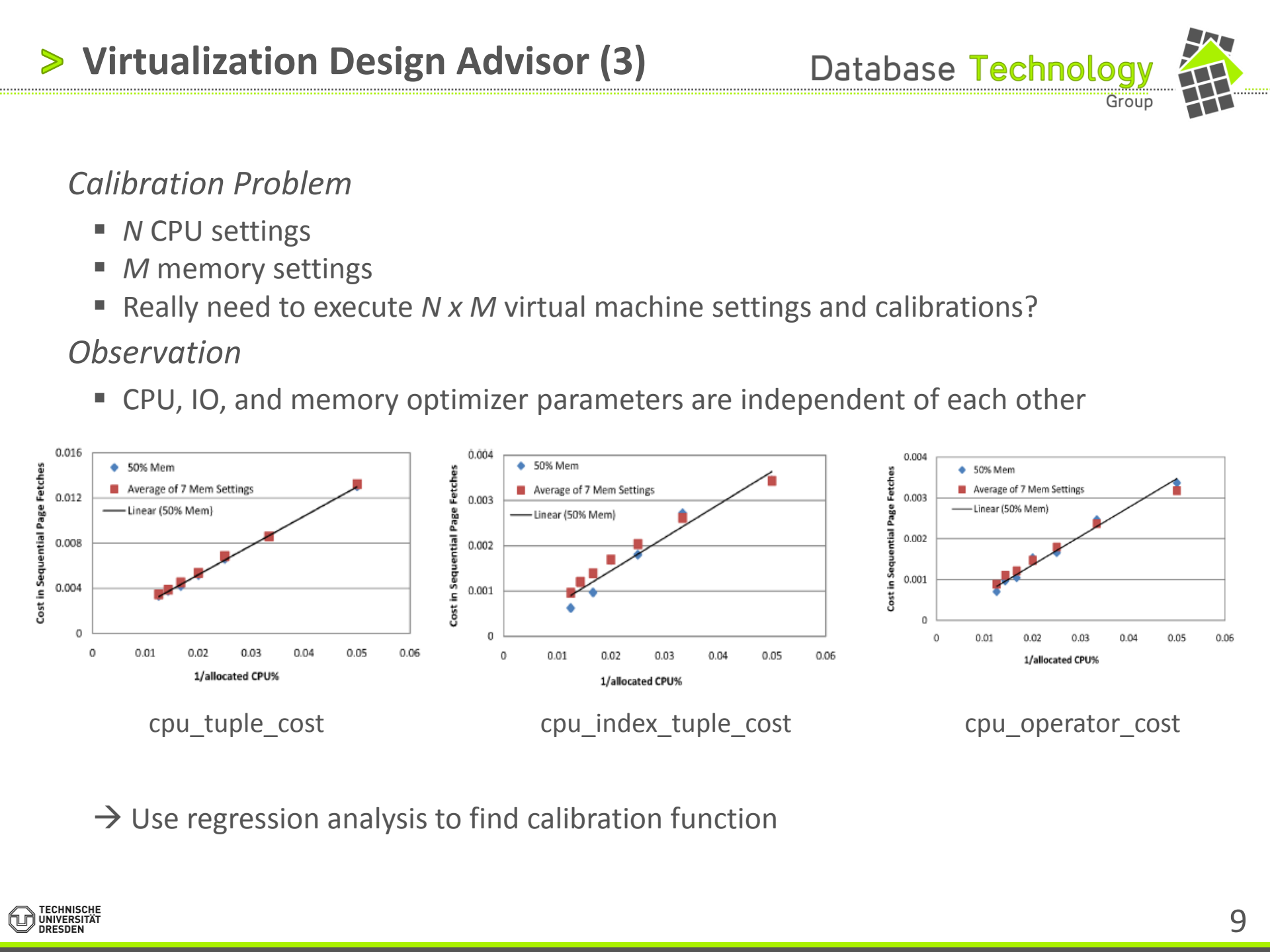
*Renormalize*

- Assumption: all DBMS defines cost as total resource consumption
- Define a mapping to unified cost estimation unit of choice (e.g., seconds)
- Highly dependent on DBMS at hand

*Calibrate*

- Prescriptive parameters
  - Define the configuration of the DBMS itself
  - E.g. `sortheap, bufferpools, …`
- Descriptive parameters
  - Characterize the execution environment
  - E.g. `cpu tuple cost, random page cost, …`
  - Affect the DBMS only indirectly (through changing cost estimates)
- Find calibration function from resource allocation $R_i$ to set of parameters $P_{ik}$

*Both tasks need to be done once for every DBMS, e.g., by DBMS developers with deep knowledge*



*Calibration Problem*

- *N* CPU settings
- *M* memory settings
- Really need to execute *N x M* virtual machine settings and calibrations?

*Observation*

- CPU, IO, and memory optimizer parameters are independent of each other




cpu_tuple_cost




cpu_index_tuple_cost

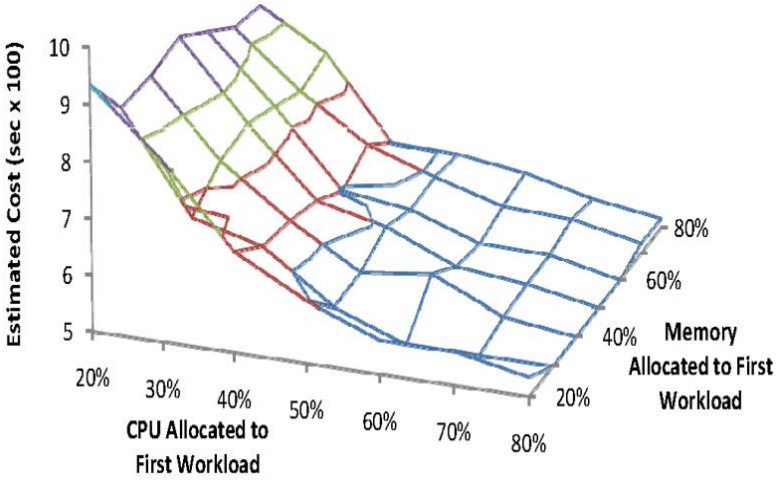



cpu_operator_cost

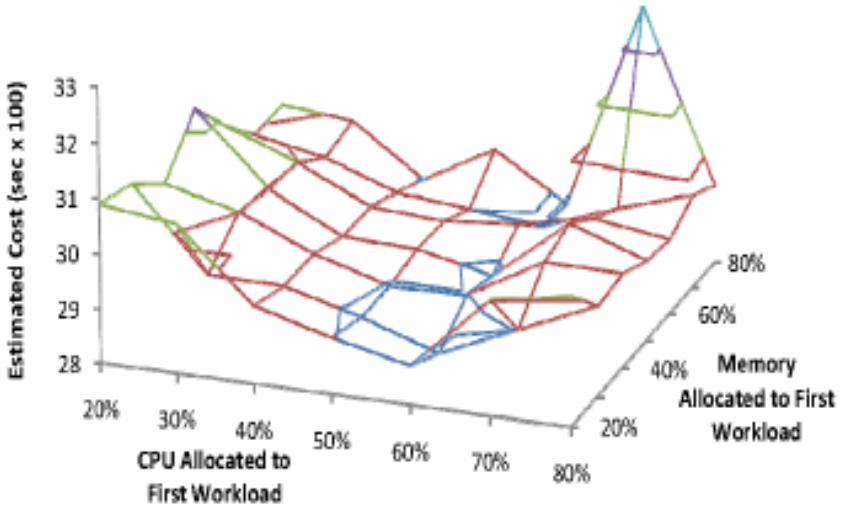
→ Use regression analysis to find calibration function

## *Configuration Enumerator*

- Greedy algorithm starting with 1/N share assignment
- Assume “concave and smooth” objective function




Two workloads not competing for CPU




Two workloads competing for CPU

## *Overhead*

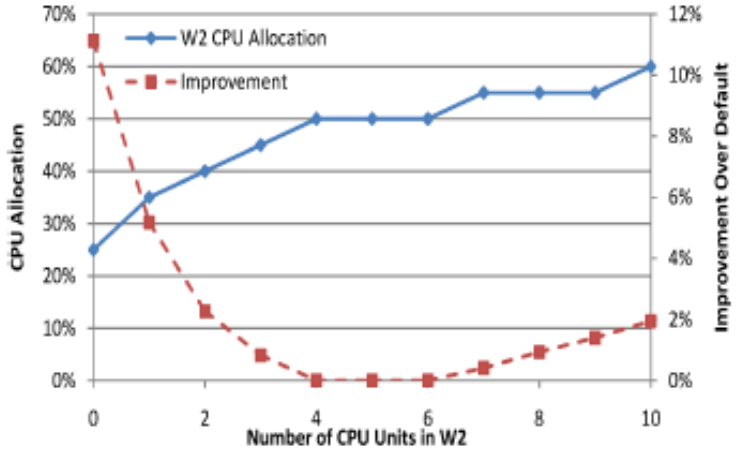
- Calibration: 9 minutes (PostgreSQL), 6 minutes (DB2)
- Enumeration: 1 minute (about 8 iterations)

*Workload Definition based on TPC-H*

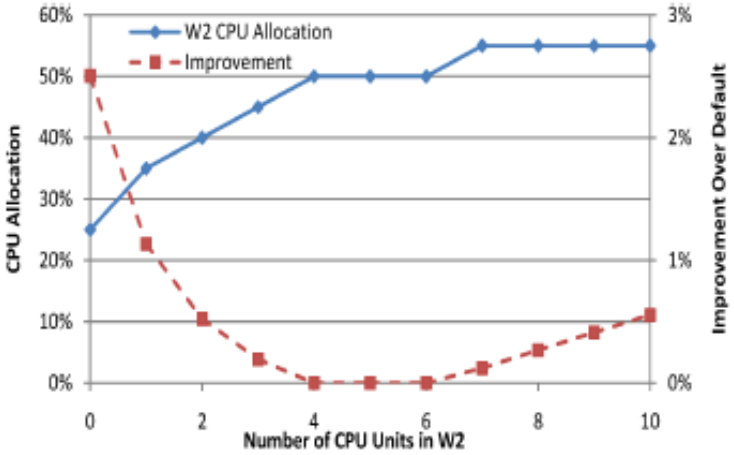
- $Q_{18}$ is one of the most CPU intensive queries
- $Q_{21}$ is one of the least CPU intensive queries (execution time much longer than $Q_{18}$)
- Workload units
  - CPU intensive workload unit = C: 25 x $Q_{18}$
  - CPU non-intensive workload unit = I: 1 x $Q_{21}$

*Experiment: Sensitivity to workload resource needs*

- $W_1 = 5 \cdot C + 5 \cdot I$
- $W_2 = k \cdot C + (10-k) \cdot I$ (increase of k → more CPU intensive)



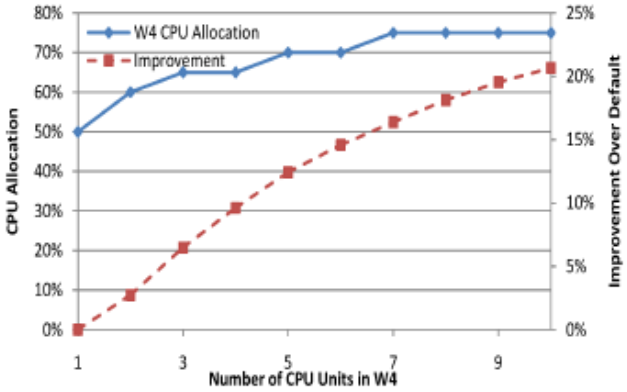

DB2



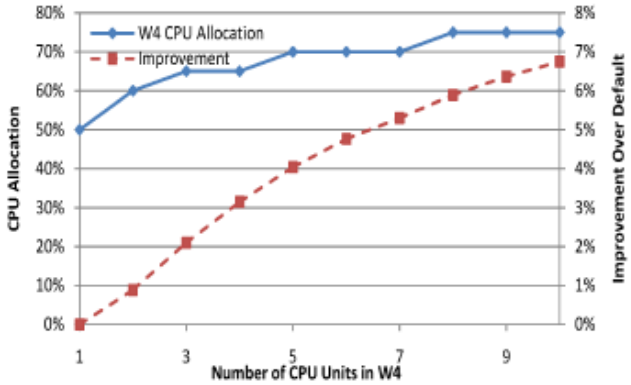

Postgres

*Workload Settings*

- $W_3 = 1 \cdot C$
- $W_4 = k \cdot C$



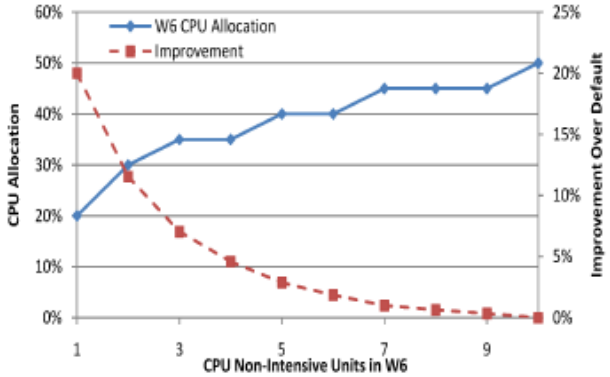

DB2



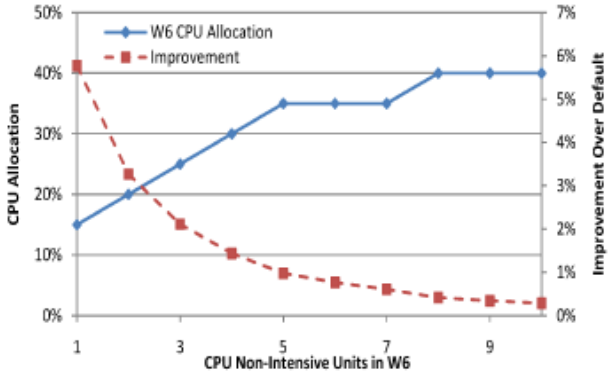

PostgreSQL

*Workload Settings*

- $W_5 = 1 \cdot C$
- $W_6 = k \cdot I$




DB2




PostgreSQL