HERCULES

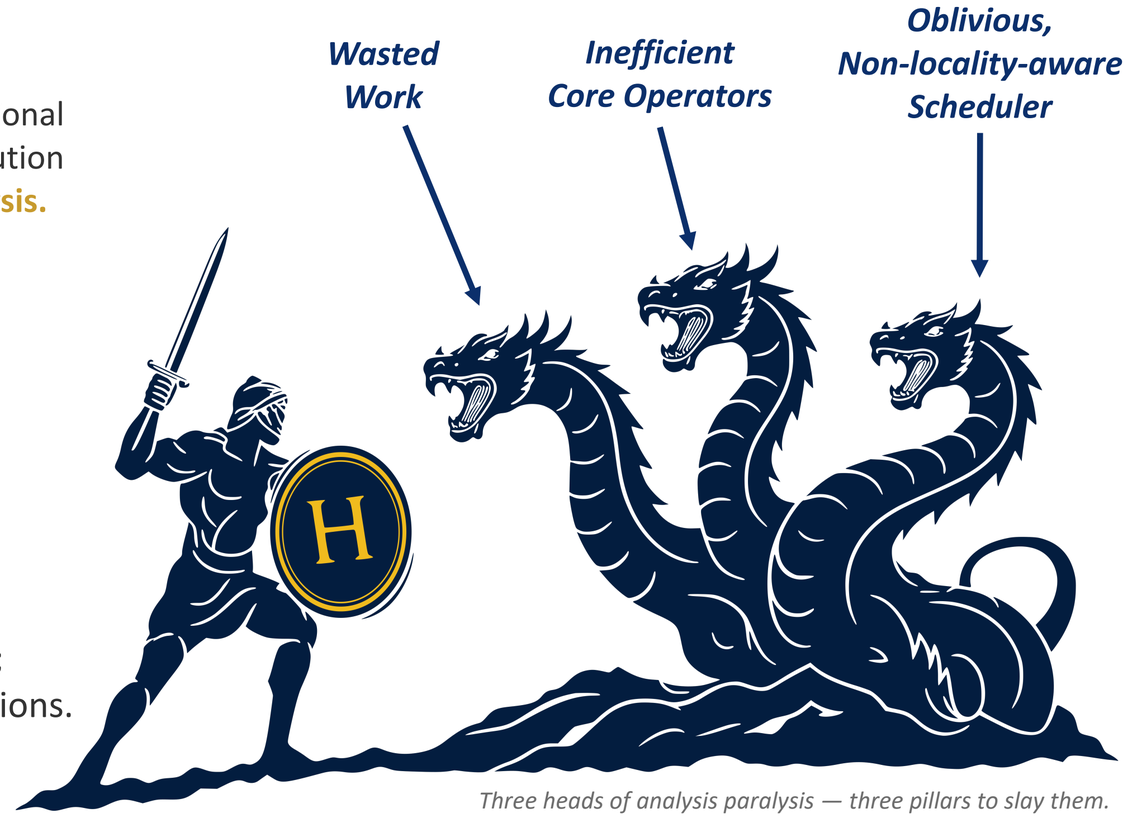
Fast Analysis, No Paralysis in GaussDB – 4.3x faster TPC-H @ 95% TPC-C rate

A. Katsarakis, V. Gavrielatos, E. Vazaios, M. Bailleu, E. Maliaroudakis, G. Stilianakis, M. Perini, M. Dananjaya, C. Reynolds, P. Guzewicz, A. Papaioannou, E. Zervakis, A. Carniel, Z. Jiahao*, Z. Pinggao*, P. Roy, N. Ntarmos
Huawei Technologies R&D (UK) – Edinburgh RC • Huawei*

PROBLEM

Modern servers should run analytics & transactions together. Today's DB engines still let analytics paralyze transactions.

Analytical processing (AP) is far from optimal in isolation. When mixed with transactional processing (TP), AP's blocking stream-based joins/aggregations and thread-intensive execution monopolize CPU and memory bandwidth, collapsing TP throughput leading to **analysis paralysis.**



Three heads of analysis paralysis — three pillars to slay them.

Three heads of the Hydra: three root causes of analysis paralysis

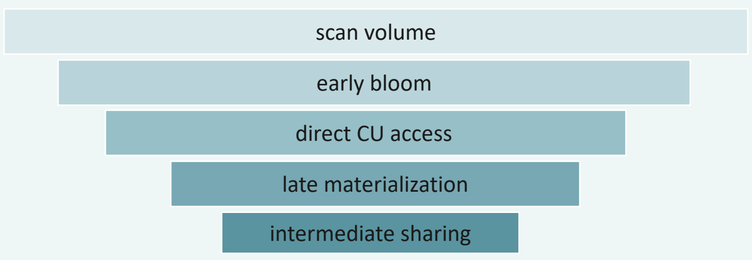
- ✗ **Wasted work:** AP incurs avoidable copies, redundant I/O, and duplicate intermediate results.
- ✗ **Inefficient core operators & sub-par concurrency:** dominant in AP hash join and aggregation execution is often neither hardware aware nor concurrency optimal.
- ✗ **Oblivious, non-locality-aware scheduler:** OS threads are not aware of DB context; poor data locality, can't suspend heavy analytics or prioritize time-critical transactions.

THE 12 LABORS

Pillar I — Fast Analysis
avoid copies and work, share what's done

1. **Early bloom filters:** build fast during runtime at hash-join build and pushed down to scans.
2. **Late reads:** filter-surviving rows carry only row-IDs & join keys until the last operator before needed.
3. **Direct CU access:** operators read directly from storage, avoiding copies to intermediate buffers.
4. **Intermediate result sharing:** similar sub-plans reuse filtered, processed, materialized outputs.

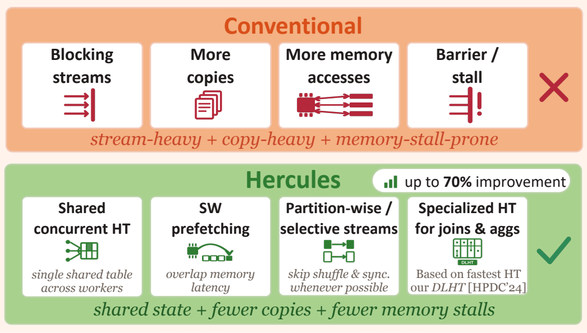
Each labor compounds its predecessor's savings.



Pillar II — Fast Dominant Operators
make joins and aggregations scale with cores, not barriers

5. **CA hash joins:** parallel build and probe over (DLHT-specialized [HPDC'24]) *concurrency-aware* hash tables; no repartition barriers or copies.
6. **CA hash aggregations:** no global shared state; redistribute when synchronization costs more.
7. **Non-blocking streams:** move data only when useful and with fewer copies and fewer stalls.
8. **Partition-wise execution:** use existing join and group-key partitions to skip shuffle & sync. costs.

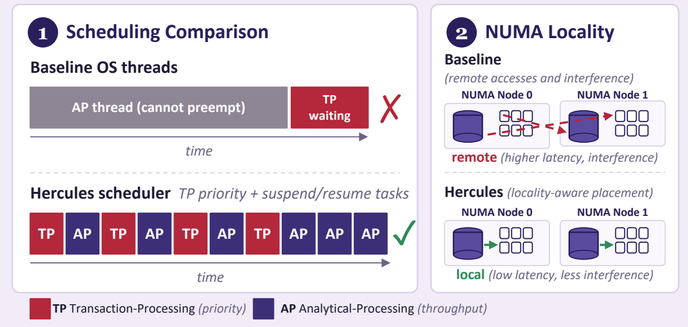
Operator tax removed: fewer stalls, mem. accesses, copies



Pillar III — No Paralysis
suspendable user-level tasks, NUMA-aware

9. **Task-based execution:** query plans are decomposed into fine-grained tasks that the scheduler can execute out of order.
10. **Suspendable:** when a task blocks on empty input or full output, it self-suspends freeing-up resources so another ready task can run.
11. **User-level scheduling:** TP tasks get priority; transactions don't wait behind heavy AP tasks.
12. **NUMA & locality-awareness:** tasks, hash tables, buffers, scan ranges stay on same NUMA node, reducing remote accesses & interference.

User-level scheduler for prioritization and locality

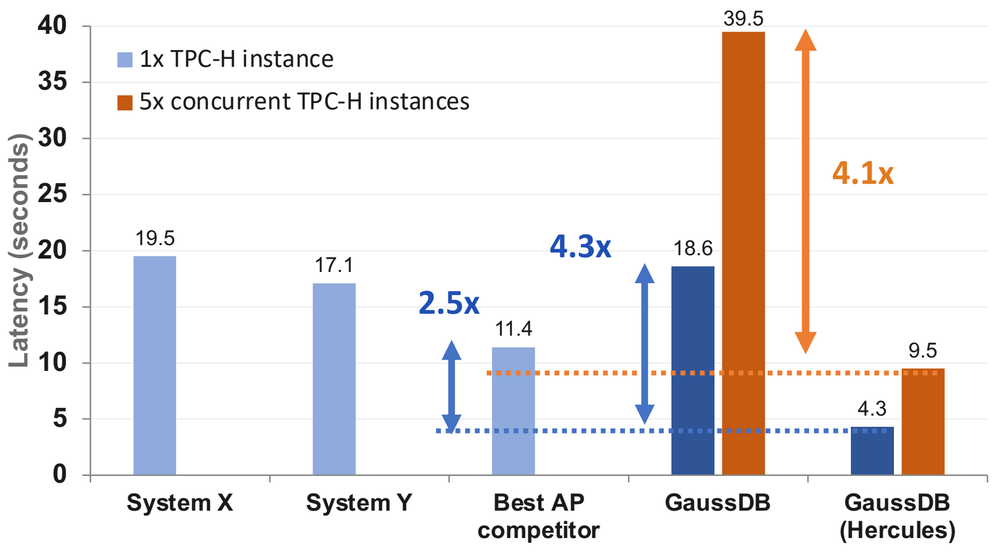


4.3x	**4.1x**	**2.5x**	**5x**	**22 sec TPC-H @ 2.4M TPC-C**
vs. GaussDB on TPC-H latency	lower latency with 5 concurrent instances	vs. best AP competitor.	concurrent TPC-H complete faster than competitors complete one	high TPC-C throughput kept while also 4x faster TPC-H.

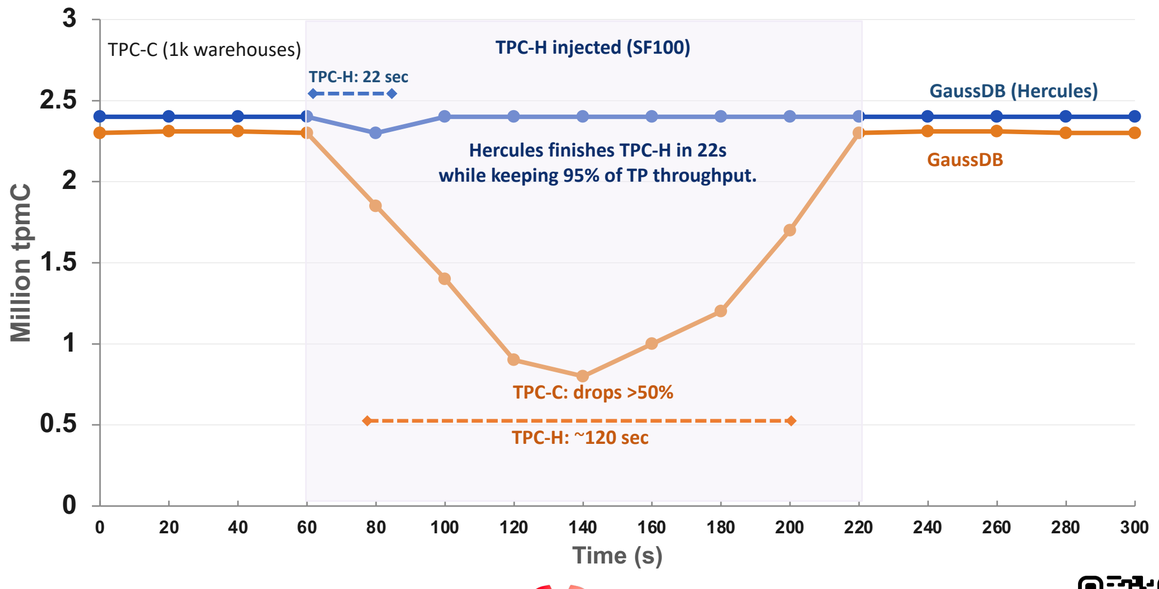
Reward

Results on a single Arm-based Huawei server (Kunpeng 920: 160 cores, 2TB DRAM)

FAST ANALYSIS: TPC-H (SF100) completion latency



NO PARALYSIS: Mixed TP-AP throughput



TAKEAWAY **Hercule's 12 labors** · twelve techniques, co-designed for modern many-core hardware to slay the Hydra of analysis paralysis!
Record-class TP/AP on a Huawei Arm server: 2.4M tpmC with TPC-H (SF100) at 22s; 4.3s AP-only

HUAWEI