LIQID

CXL Memory
Solution Brief

Why Liqid

LIQID enables enterprises to dynamically pool and scale memory across servers, eliminating bottlenecks while improving throughput, efficiency, and overall database performance.

Key Advantages

» Higher throughput with in-memory datasets

» Lower latency from reduced data movement

» Eliminate stranded DRAM with memory pooling

» Cut costs and scale via CXL, not new servers

www.liqid.com

Memory Centric Acceleration with CXL

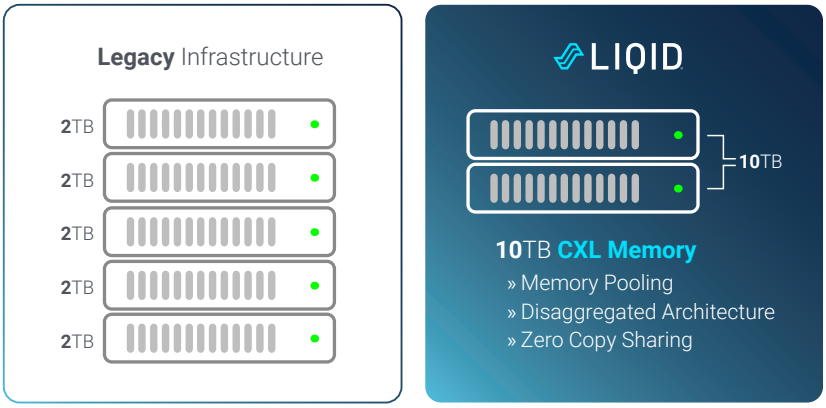
Enterprise performance is now **memory bound**. DRAM is fast but capacity limited and expensive; scaling by sharding adds latency, complexity, and cost. **Compute Express Link (CXL)** changes the equation by decoupling compute from memory so you can **expand and pool memory across servers** with near DRAM characteristics. **Liqid** delivers the fabric and **software defined orchestration** that operationalizes CXL in production, so you can keep dramatically larger datasets in memory, unify AI and database workloads, and hit SLAs with fewer nodes.

The Problem Liqid Solves

- **DRAM ceilings** force sharding and overprovisioning just to meet peak memory demand.
- **Latency tax** from remote lookups, network hops, and constant data movement between tiers and systems.
- **I/O bound analytics** where scans/joins thrash SSDs instead of running from memory.
- **Siloed AI + data stacks:** Copying embeddings/tables between CPU and GPU memory wastes time and capacity.
- **Stranded capacity and rigid boxes:** Memory is trapped per server; upgrades require disruptive refreshes.

The Liqid Approach

- **CXL Memory Pooling & Expansion:** Create a **shared pool of memory** that any host can draw from on demand; scale memory independently of CPU.
- **Tiered Memory, Not Just Bigger DRAM:** Keep **hot** pages in DRAM and **warm/cold** data in CXL.mem for the best price/performance.
- **Coherent Access for CPUs & GPUs:** Enable AI inference/training and databases to operate on the **same memory pool** without expensive copies.
- **Policy Driven Composability:** With **Liqid Matrix**, discover resources, compose per workload memory footprints, and **recompose in minutes** as priorities shift.
- **Future Ready:** A clear path toward **persistent CXL** to accelerate checkpoint/recovery and simplify durability models.



Benefits and Outcomes

- **Performance at lower node counts:** Keep far larger datasets resident in memory; reduce sharding and coordination overhead.
- **Lower Cost to Performance:** Stop overbuilding every box for peak DRAM; pool memory where it's needed, when it's needed.
- **Operational Agility:** Rebalance memory for quarter end, product launches, or new models **without re-racking**.
- **AI + Data Convergence:** Run OLTP/HTAP, vector search, and model inference against the **same in memory substrate**.
- **Resilience Roadmap:** Prepare for persistent CXL to shrink recovery windows and simplify logging/checkpointing.

Priority Workloads

- **Vector / RAG indexes:** Keep larger embeddings and IVF/HNSW structures in memory; cut top K latency.
- **Graph analytics & fraud:** Irregular pointer chasing thrives when long tail edges sit in CXL.mem while hot subgraphs stay in DRAM**.**
- **In memory analytics & HTAP:** Turn scan heavy queries from I/O bound to memory bound; speed joins/group bys.
- **OLTP + AI side by side:** Transactional systems and inference share the same coherent pool—no copies.
- **Time series & observability:** Retain longer histories in memory for faster rollups and anomaly detection.

How It Works Step by Step

1. **Discover:** Liqid Matrix inventories hosts, CXL memory devices, GPUs, and NVMe over standard fabrics.
2. **Compose:** Allocate the right memory footprint per workload, with guardrails and tenancy controls.
3. **Run:** Databases and AI jobs access tiered memory (DRAM + CXL.mem) as if local, minimizing I/O and copies.
4. **Monitor:** Track hotness, utilization, and SLA adherence.
5. **Recompose:** Shift memory between hosts in minutes as demand changes, no app refactoring.

Reference Architectures at a Glance

- **Scale Up Node with Tiered Memory:** 1–2 sockets + DRAM for hot data, **CXL mem** for warm/cold; optional GPU pool attached via Liqid fabric.
- **Converged AI + Data Host:** OLTP/HTAP or vector DB shares a memory pool with GPUs for recommendation/LLM inference.
- **Memory Pool for a Small Cluster:** 3–6 hosts drawing from a shared CXL pool; each host composes what it needs for the hour/day.

Getting Started

1. **Identify Memory Bound Candidates:** Vector/graph/HTAP/time series with frequent spills or high cache miss rates.
2. **Right Sizing Study:** Measure working set size, access skew (hot vs. warm), and I/O hotspots.
3. **Pilot:** Attach a Liqid enabled CXL pool; compose tiered memory to 1–2 hosts; validate SLA, node count, and I/O reduction.
4. **Scale Plan:** Roll out policies for hotness aware tiering and multi tenant allocation; expand pool capacity as needed.

Why Liqid

- **Operationalizes CXL** on standard servers with vendor agnostic flexibility.
- **Software defined control plane** to compose GPUs, memory, and NVMe with repeatable, policy driven workflows.
- **Proven integrations** with common schedulers/orchestrators to fit existing ops.

Call to Action

Ready to quantify the gains? Run a pilot to measure latency reductions, node consolidation, and cost to performance improvements on your stack.

www.liqid.com