Lovelock: Towards Smart NIC-hosted Clusters *

Seo Jin Park
University of Southern California

Ramesh Govindan
University of Southern California

Kai Shen
Google

David Culler
Google

Fatma Özcan
Google

Geon-Woo Kim
UT Austin

Hank Levy
Google
University of Washington

Abstract

Traditional cluster designs were originally server-centric, and have evolved recently to support hardware acceleration and storage disaggregation. In applications that leverage acceleration, the server CPU performs the role of orchestrating computation and data movement and data-intensive applications stress the memory bandwidth. Applications that leverage disaggregation can be adversely affected by the increased PCIe and network bandwidth resulting from disaggregation. In this paper, we advocate for a specialized cluster design for important data intensive applications, such as analytics, query processing and ML training. This design, Lovelock, replaces each server in a cluster with one or more headless smart NICs. Because smart NICs are significantly cheaper than servers on bandwidth, the resulting cluster can run these applications without adversely impacting performance, while obtaining cost and energy savings.

ACM Reference Format:
Seo Jin Park, Ramesh Govindan, Kai Shen, David Culler, Fatma Özcan, Geon-Woo Kim, and Hank Levy. 2024. Lovelock: Towards Smart NIC-hosted Clusters . In *Proceedings of 3rd Workshop on Sustainable Computer Systems (HotCarbon'24).* ACM, New York, NY, USA, 8 pages.

1 Introduction

Until recently, datacenter clusters were server-centric: servers with big compute and storage, connected by a high-speed fabric, enabled massively parallel data processing applications. In these, a single application instance can recruit tens of thousands of worker nodes to load and process input data in parallel, followed by shuffling results through the network fabric. For large datasets, such applications can consume massive computational power and bandwidth.

More recently, cluster designs have evolved to accommodate *acceleration* and *disaggregation*. Custom hardware can be more efficient for some workloads (e.g., ML training and inference, video encoding/decoding), so cluster designs now include accelerators attached to servers. Clusters also disaggregate storage — dedicated servers for serving storage requests over the network — in an effort to scale storage and compute independently. Recent research suggests that future clusters will disaggregate memory [40, 41, 47] and accelerator access [10, 22] as well to circumvent the problem of right-shaping resources to tasks.

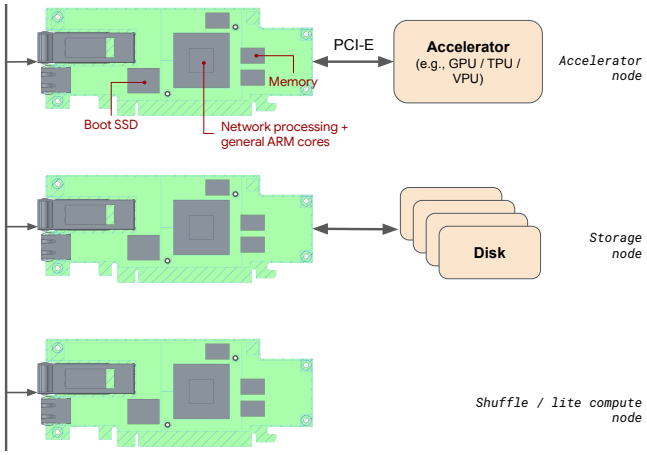

Figure 1: Architecture of Lovelock

In this paper, we take the position that with the advent of acceleration and disaggregation, for several important applications (analytics, query processing, ML training), a server-centric design may no longer be necessary (§2). In servers with increasing core counts, cores contend for network, memory and PCIe bandwidth. This is exacerbated by disaggregation, which increases traffic on the PCIe bus and the network. For applications that use accelerators heavily, the server CPU is reduced to the role of a coordinator, merely orchestrating computations and data movement to avoid accelerator stalls.

Instead, we argue that it is more cost-effective and energy-efficient to design specialized server-less clusters for these applications. Our proposed cluster design, Lovelock, replaces servers with headless smart NICs (Figure 1). Today's smart NICs (e.g., Intel IPU E2000 [2], Bluefield DPU [6], AMD Pensando [1]) were originally designed

*The system name is after English scientist and environmentalist, James Lovelock.

HotCarbon'24, July 9, 2024, Santa Cruz, CA

	Cores vCPUs	NIC	DRAM	NIC bw per core	DRAM bw per core
Cloud host systems					
Google Cloud N1 2x Intel Skylake	96	100Gbps	2x 6-ch DDR4	0.13 GB/s	2.67 GB/s
Google Cloud N2d 2x AMD Milan	224	100Gbps	2x 8-ch DDR4	0.06 GB/s	1.83 GB/s
AWS M6in 2x Intel Ice Lake	128	200Gbps	2x 8-ch DDR4	0.20 GB/s	3.20 GB/s
Google Cloud C3 2x Sapphire Rapids	176	200Gbps	2x 8-ch DDR5	0.14 GB/s	3.49 GB/s
AMD Genoa[1] (1x EPYC 9654)	192	200Gbps	12-ch DDR5	0.13 GB/s	2.40 GB/s
Smart NICs					
IPU E2000 [43]	16	200Gbps	3-ch LPDDR4	1.56 GB/s	6.40 GB/s
Bluefield v3 [5]	16	400Gbps	2-ch DDR5	3.13 GB/s	5.60 GB/s

[1] "AMD Genoa" is not yet released on public clouds, so we assumed 1 socket of EPYC paired with a 200Gbps NIC and the highest possible memory bandwidth.

Table 1: Network and DRAM bandwidths per core of different platforms. The reported bandwidths are theoretical, not effective bandwidths by measurements. The theoretical DDR bandwidths were computed using DDR transfer rates if reported publicly, or the max transfer rate of the respective DDR technology otherwise.

to offload networking and infrastructure tasks, but they possess enough compute (e.g., 16 ARM cores), memory (16-48 GBs) and PCIe connectivity to serve as a platform for disaggregation (§2). A smart NIC also costs substantially less in capital and operating (energy) costs — e.g., $1500 vs. $10500 (7x) and 65W vs. 728W (11x) respectively in [6]. Thus, even if a Lovelock cluster were to replace each server with *multiple* smart NICs, it could still be substantially cheaper and more energy-efficient than a server-centric design.

The concept of clustering Smart NICs is not new. Sirius [11] demonstrates the offloading of stateful network functions to a cluster of Pensando NICs, where Smart NICs still serve the original purpose, improving networking performance. In contrast, our proposal involves offloading select user applications to a Lovelock cluster, where we exploit the high bandwidth of Smart NICs for bandwidth-hungry data-intensive workloads.

Lovelock can improve efficiency without compromising the performance of data-intensive applications because smart NICs offer substantially higher network and memory bandwidth-to-compute ratios than traditional servers (Table 1). The high network bandwidth enables faster network transfers for applications that leverage disaggregation, compensating for the lower CPU speeds on the smart NIC (§5). The higher memory bandwidth allows each core of a smart NIC to be more efficient, relative to a server core. Using a simple model of cost and power (§4), we show that for certain applications, Lovelock can reduce capital cost by 21%-71% and energy use by 23%-80%.

These preliminary, back-of-the-envelope analyses are encouraging, but require significant work in improving the design of smart NICs, increasing the efficiency of the network stack and isolation mechanisms, and scaling disaggregated applications efficiently (§6).

2 Background and Motivation

We begin with a brief background on smart NICs, then make several observations that motivate our work.

2.1 Smart NICs

Originally, smart NICs were designed to offload packet processing from the host CPU with the goal of preserving CPU cycles for application workloads. Early smart NICs, for example, supported TCP segmentation and re-assembly, measurement, and access control, and could also be programmed to perform general packet match-action tasks [3, 4]. Since then, smart NICs have evolved to have on-board general-purpose compute and significant memory, to the point where they are considered generalized data processing units (DPUs) or infrastructure processing units (IPUs).

A latest example of such smart NICs is IPU E2000 from Intel. It has a built-in processor with 16 Arm cores and a low-power DRAM (LP-DDR4). On this hardware, today's smart NICs run a commodity operating system (such as Linux), and can support power-efficient execution of general purpose computations without requiring significant code modifications. Beyond these, smart NICs have specialized hardware for common tasks. For example, the E2000 has a programmable match-action packet processing pipeline to implement access control, NAT, or congestion control in hardware. Smart NICs commonly support acceleration for encryption and compression, two operations that consume significant CPU cycles in datacenters [21]. These accelerators free up the use of compute cores for other tasks, a capability we exploit in this paper. It is also common for smart NICs to provide PCIe connectivity to attach accelerators, storage, and other peripherals.

2.2 Motivation

Several trends in datacenter computing and data-intensive applications motivate our work.

Increasing core counts create bandwidth bottlenecks. Cloud operators sell CPU cores (accompanied with 4 GB or so DRAM per core) to customers. To reduce per-core capital cost, it is now common for a host system to have hundreds of cores. Consequently, the system network and memory bandwidths are now shared with more cores, which have increasingly bottleneck-ed application performance.

Weak isolation and its impact on tail latency. To utilize hundreds of cores, a host now has to serve multiple independent applications (or cloud VM instances) [12, 46]. Applications are typically assigned dedicated CPU cores and some reserved memory capacity. However, other resources, such as memory bandwidth, last level cache, PCIe bandwidth, and network bandwidth, are still shared. Contention on those shared bandwidth resources can degrade application performance. This can potentially be alleviated using class-of-service or QoS enhancements to some of these resources (e.g., ToS for network traffic, and isolation mechanisms for other shared resources [13, 18, 33]), but in practice, these provide weak isolation. Even mild contention can result in higher tail latency and worse end-to-end performance especially for data-intensive workloads targeted in this paper.

Disaggregation increases PCIe and network traffic. Disaggregating memory and storage help independently scale storage and computation, and can increase memory and storage efficiency. However, disaggregation can add significant traffic to the network and the host-to-NIC PCIe bus. On a disaggregated host, memory traffic

must traverse the PCIe bus and the network. Similarly, disaggregated storage traffic consumes additional network bandwidth, and additional PCIe bandwidth at the remote end. Increasingly, the PCIe bus is becoming a significant bottleneck for applications [8, 35]. This is exacerbated by the increase in host-attached accelerators for graphics, video, and machine learning.

The changing role of the host CPU. With increased use of hardware acceleration, the role of the CPU on a server in a data center has been changing. Increasingly, the CPU runs application logic that rarely performs intensive computation but focuses on coordinating computation on the accelerators and on transferring data between these devices and disaggregated memory and storage to avoid stalls on accelerators.

3 Lovelock: Clusters for Data-intensive Workloads

Motivated by these trends, we explore a novel architecture, Lovelock, for a specialized pod or cluster for some data-intensive workloads. A Lovelock cluster is distinguished by the complete absence of server-class machines (Figure 1). Instead, in Lovelock, smart NICs perform the functions of servers in a traditional cluster. Thus, the cluster consists entirely of network-attached smart NICs.

In addition, each smart NIC may have one or more additional peripherals connected over PCIe, such as accelerators and SSDs. Specifically, we envision each node in a Lovelock cluster to be one of: an *accelerator node* which contains an attached GPU, TPU, video processor, crypto accelerator, etc; a *storage node* that contains several physical storage devices (e.g., SSDs or HDDs) and serves storage requests over the network; or a *lite compute* node without peripherals used entirely for lightweight computations or data shuffles.

Lovelock is a specialized architecture for a specific subset of applications (bandwidth, not compute, bound applications) that leverages the potential cost and power benefits that smart NICs provide. It leverages the trends described in Section 2.2 as follows:

- Per-core memory bandwidth and shared cache are larger in Lovelock, resulting in higher per-core performance relative to cores on traditional servers.
- Each smart NIC now serves fewer (or a single) applications, lessening the chance of contending on shared network/memory/PCIe bandwidths.
- Lovelock improves disaggregation by having higher network bandwidth and removing the PCIe traffic between NIC and host CPUs. For example, the IPU E2000 uses a special mesh fabric, instead of PCIe, between the network processor and its ARM cores.)
- For applications in which the CPU simply acts as a coordinator, the minimal compute on DPUs in Lovelock is a better fit in terms of power consumption.

Because a smart NIC can be an order of magnitude cheaper and more power efficient than a traditional server, a Lovelock cluster can *scale out* smart NICs — replace one server with multiple smart NICs — to achieve comparable application performance while still being more cost and energy efficient (§4). This scale out results in a cluster with higher aggregate bandwidth, which can benefit some applications (§5).

In this paper, we take a first step towards understanding the feasibility of Lovelock. Specifically, we:

- Explore, using very simple analytic models, the cost and power-efficiency gains from Lovelock relative to traditional clusters with servers (§4)
- Describe, and substantiate with measurements, a few applications that can benefit from Lovelock (§5).
- Discuss directions for future research (§6).

4 Energy and Cost Modeling

The cost and energy benefits of Lovelock are somewhat difficult to quantify, in part due to the scarcity of public information on capital costs, and because both cost and power advantages can change over time. We use an analytical model to get a preliminary understanding of Lovelock's benefits. Our analysis is best-effort given the available public information.

Notation. Suppose c_s is the capital cost of a server relative to that of a SmartNIC and p_s the power draw of a server relative to a SmartNIC. Analogously, let c_p be the cost of PCIe devices (again, relative to the SmartNIC) attached to a server in a traditional cluster, or to the SmartNIC in Lovelock, and let p_p be their relative power. Now, a Lovelock cluster is likely to be slower than a traditional cluster, which presents cluster designers with two degrees of freedom: they can provision ϕ times more SmartNICs than a traditional cluster servers and/or accept a slow-down μ on application execution. These two terms are knobs that designers can use to trade-off cost, power, and application performance[1].

Cost and energy saving. Using the notation above, we can approximate the ratio of the capital cost of a traditional cluster to the cost of a Lovelock cluster as:

$$\frac{c_s + c_p}{\phi + c_p} \tag{1}$$

and the ratio of the power draw of a traditional cluster to that of a Lovelock cluster as:

$$\frac{p_s + p_p}{\mu(\phi + p_p)} \tag{2}$$

A recent white paper from NVIDIA on their Bluefield v2 SmartNIC [6] suggests $c_s \cong 7$ and $p_s \cong 11$. A Lovelock cluster without PCIe devices that runs bandwidth-intensive applications and has 3× as many SmartNICs as servers (i.e., $\phi = 3$) and runs these applications 20% slower (i.e., $\mu = 1.2$) is still 2.3× cheaper and uses 3.1× less energy!

For a cluster with PCIe devices, assume that the cost and power of PCIe devices is about 75% of the total system[2]. Then, using $c_s = 7$, $p_s = 11.2$ in [6] again, the cost and power ratios for PCIe devices will be $c_p = 7 \times \frac{0.75}{1-0.75} = 21$ and $p_p = 11.2 \times \frac{0.75}{1-0.75} = 33.6$. A Lovelock cluster with 1 smart NIC in place of 1 server (i.e., $\phi = 1$) and without any slowdown, has a 1.27x cost saving and 1.3x energy reduction. If Lovelock is configured to use 2x more smart NICs ($\phi = 2$) to improve application performance by 10% ($\mu = 0.9$), it can save 1.22x on cost and 1.4x on energy.

In §5, we use this model to quantify benefits of Lovelock.

[1] For ease of exposition, we have omitted fabric costs from the model. However, the model can be extended easily to account for increased fabric costs; we discuss this in §5.2 and §6.

[2] Rough estimate based on commercial systems with 4 GPUs/server.

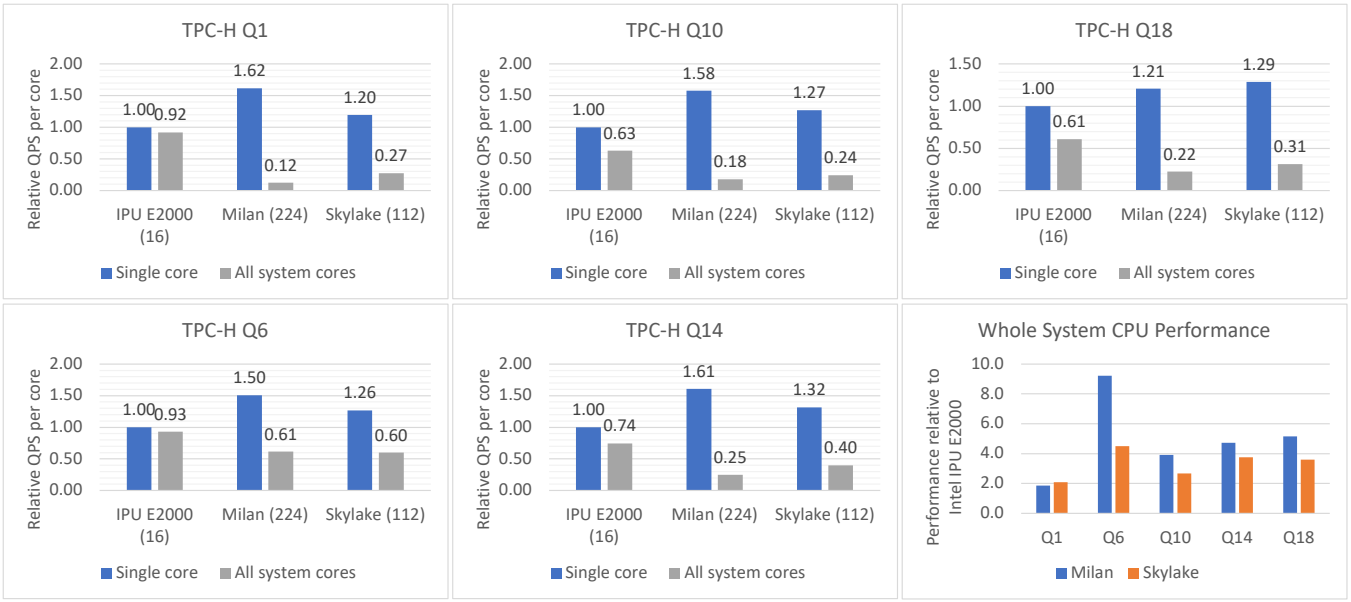

Figure 2: Per-core performance when each core (SMT) independently executes a TPC-H query (so, no synchronization among cores). A proprietary analytics execution engine and TPC-H scale factor 1 (about 1 GB of data when uncompressed) were used. Performance was measured by execution time and normalized by the performance of Intel IPU E2000 when used only one core.

5 Initial Study Results

In this section, we explore the following hypotheses with respect to Lovelock clusters:

- Smart NIC CPU cores can outperform traditional hosts for memory-bandwidth-intensive workloads (§5.1).
- Higher network bandwidth can improve query processing performance at lower cost (§5.2).
- They have CPU and memory capacity to drive high performance accelerators such as GPUs/TPUs, and giving higher bandwidth per accelerator reduces accelerator stalls (§5.3).

5.1 Higher CPU Core Efficiency

Smart NICs have 7-11x fewer cores than traditional systems (Table 1). If a smart NIC is ~7x cheaper than a traditional host [6], a Lovelock cluster with compute capacity comparable to a traditional cluster will have no cost advantages.

However, we anticipate that, at least for data-intensive workloads, each core of smart NIC can outperform a traditional host's core because it has higher memory bandwidth and larger L3 cache. To quantify this, we run TPC-H benchmarks with scale factor of 1 on an analytics execution engine to show that contention on shared bandwidth impacts traditional host core performance much more than a Smart NIC core.

We use three different systems for this evaluation. *IPU E2000* has 16 ARM N1 cores and 48 GBs of memory. *Milan* (same as Google Cloud N2d) has 224 AMD Milan SMTs and 1.83 GB/s memory bandwidth per SMT. *Skylake* (same as Google Cloud N1) has 112 Intel Skylake SMTs (2 sockets of 28 cores), 2.3 GB/s memory bandwidth per SMT.

Figure 2 shows the per-core performance when all cores independently run identical TPC-H query executions concurrently. For reference, we also measured the query execution performance when only one core is busy. When we benchmark systems with a single thread, the performance of AMD Milan and Intel Skylake is higher than that of the Smart NIC. When all cores run independent TPC-H query executions concurrently, the per-core performance of Intel IPU E2000 drops by 8–39% (16 cores total). On the other hand, the per-core performance of x86 systems drops by 39%–88%. Across all cores on each system, AMD Milan shows 1.9-9.2x (median 4.7x) performance of E2000, and Skylake is 2.1-4.5x (median 3.6x) that of E2000. This suggests that a Lovelock cluster with a ϕ of 3.6-4.7 might suffice to match the CPU performance of traditional servers.

The lone exception, the TPC-H Q6 query, performs a compute-bound scan of data in memory. The performance of Milan and Skylake drops mostly due to SMT core sharing.

5.2 Higher End-Host Network Bandwidth

Relative to a traditional cluster, an important advantage of a Lovelock cluster with $\phi > 1$ is the higher aggregate end-host network bandwidth due to more Smart NICs. For big data workloads that involve large network transfers, a Lovelock cluster can be cheaper and more energy-efficient: it can speed up network transmission to compensate for computation slowdown as a result of lower aggregate compute power.

A recently published breakdown of Google's BigQuery processing time [21] reports that, on average, over 60% of total time is spent on network operations, mainly remote shuffle and disaggregated storage IO. Using this breakdown, Lovelock with $\phi > 1$ will provide higher network bandwidth, potentially reducing the remote shuffle and IO time.

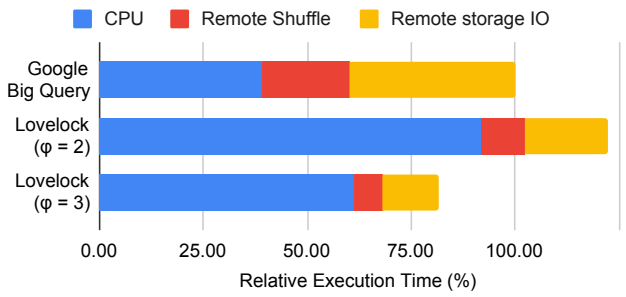

Figure 3: Prediction of Big Query execution time with Lovelock, based on the profiling data in [21].

Figure 3 projects changes in BigQuery processing time with Lovelock. To project CPU time, we multiply by 4.7, the median value of Milan's whole system CPU performance relative to E2000 in Figure 2; then, we divide by ϕ since we assume linear speedup. For remote shuffle and storage I/O time, we assume they are bottlenecked by network bandwidth. This is reasonable since BigQuery jobs usually scan terabytes or more of data, so shuffle and I/O involve large data transfers. Following [21], we attribute RPC processing at BigQuery workers to CPU times, not network transfers.

The first row in Figure 3 corresponds to the execution time composition reported in [21]. We present two Lovelock configurations: 2x and 3x more NICs than traditional servers (i.e., ϕ is 2 or 3). With $\phi = 2$, total execution time increases by 22% ($\mu = 1.22$) because network overhead reduction is not enough to fully compensate $\frac{4.7}{2} = 2.35$x reduction on aggregate CPU performance. With $\phi = 3$, total execution time will reduce by 19% (i.e., $\mu = 0.81$).

For these two configurations, our model, together with cost and power values from [6], suggests that Lovelock's device cost advantage is 3.5x (respectively 2.33x) for ϕ of 2 (respectively 3). The energy savings are 4.58x for both.

Our model (§4) ignores networking cost (fabric and ToR) increases for supporting more NICs. If we assume that networking accounts for 10% of traditional cluster, Eq. 1 can be extended to $\frac{c_s+c_f+c_p}{\phi\cdot(1+c_f)+c_p}$, where c_f is the networking cost and may be assumed to be $c_s \times 10\% = 0.7$. With this updated cost model, the cost benefits with $\phi = 2$ and $\phi = 3$ will be 2.26x and 1.51x, respectively. We discuss this more in §6.

However, this analysis is pessimistic since it assumes fabric costs scale linearly with ϕ. Instead, fabric capacity needs to increase only to keep up with execution time as determined by the slower CPUs. Thus, with $\phi = 2$, the application slows down by $\mu = 1.22$, so the fabric can actually be slower by about 19% ($1 - \frac{100\%}{122\%}$). Similarly, for $\phi = 3$, the fabric needs to be faster by about 23% to sustain the performance speedup. Thus, to sustain $\phi > 1$, it may not be necessary to provision ϕ times more capacity; rather it may be sufficient to over-subscribe the network.

Model	Mean CPU%	Peak CPU%	Model Size (per accel/Host)	Mean Mem Use	Max Mem Use
GLaM1B	4.8%	8.9%	0.2GB / 0.8GB	3.4GB	5.0GB
GLaM4B	3.8%	6.2%	0.4GB / 1.8GB	3.8GB	6.5GB
GLaM17B	3.4%	10.2%	2.0GB / 8.1GB	4.2GB	17.8GB
GLaM39B	2.1%	13.3%	4.5GB / 18.2GB	4.7GB	35.7GB

Table 2: Host CPU and DRAM use during distributed training. "CPU%" is normalized to the IPU E2000's CPU performance. CPU and memory use are sampled every minute from all 8 hosts, and avg and peak are calculated from the sampled data over the whole training.

5.3 Ability to Drive Accelerators

Lovelock can benefit accelerator-based workloads in which (a) the CPU coordinates accelerator execution and data movement, and (b) accelerators are network-bound.

CPU as coordinator. In large language model training, CPUs effectively only coordinate training. To demonstrate that Lovelock can lower the cost of this training, we trained large language models on 8 hosts each of which has 4 ML accelerators that can individually deliver about 50 TFLOPs. We used multiple model sizes, ranging from 1B to 39B, based on the configuration of dense models used in GLaM [16]. The model parameters were evenly partitioned across the accelerators, and we set a global batch size of 64. With this training setting, we measured the resource usage of the hosts for 1,000 training steps. The role of CPU in this workload ranges from dispatching tasks to accelerators, checkpoining, and moving data across the network. The workload uses both inter-accelerator interconnect and datacenter network.

Table 2 shows the CPU and memory usage in host machines. Even the peak CPU use is well below the capacity of a smart NIC, IPU E2000. On average, training consumes only 3-5 GBs of memory, well below the capacity of a smart NIC. However, peak memory consumption can go up to twice the model size, when checkpointing the current training snapshots, including model parameters and optimizer states. We believe it is possible to reduce this peak by splitting model parameters into chunks and checkpointing a stream of these chunks. With this change, since an IPU E2000's DRAM capacity can be configured up to 48 GBs, each E2000 can drive 2-4 accelerators depending on the model size.

Thus, Lovelock with $\phi = 1$ can likely sustain LLM training without any performance slowdown. Assuming that the device and energy cost of a host is 25% of the entire system – based on current servers with 4 GPUs – and using cost and power values from [6] ($c_s = 7$, $p_s = 11.2$, $c_p = 21$, and $p_p = 33.2$), Lovelock's cost advantage is 1.27x, and energy savings is 1.30x.

Higher aggregate network bandwidth. Graph Neural Network (GNN) training is network bandwidth intensive. GNNs generate node embeddings from graph-structured datasets [14, 24, 28, 45]. GNN computation requires significant network communication to preserve data dependencies in graphs [19, 29, 32]. For example, recent work [32] shows that creating one mini-batch requires fetching 200MB data from remote machines. While 8 V100 GPUs in one machine can compute 400 mini-batches per second, the shared 100Gbps network only allows 60 mini-batches, resulting in accelerator stalls and under-utilizing accelerators.

Such stalls can also occur more generally in synchronous data-parallel training and model-parallel training/inference. Even if network bandwidth is provisioned enough to support average throughput, accelerators can still stall waiting for network transfers to complete. Such network stalls often account for over 20% of execution time [34, 36], so providing 2x of bandwidth can easily bring 10% speedup. A Lovelock cluster with $\phi = 2$, assuming accelerators account for 75% of system power and cost (§4), will have 1.22x cost and 1.4x power advantage over a traditional cluster.

6 Discussion and Future Work

Improving Smart NICs for Lovelock. Some smart NICs have limited memory bandwidth because their CPUs were designed to handle only subset of workloads. For example, Bluefield v3 has a memory bandwidth that is only 1.8x of network bandwidth (Table 1), so the internal CPU cannot process the data at line rate (IPU E2000 doesn't exhibit this limitation). Future NICs for Lovelock can either allocate higher memory bandwidth or support DMA to PCIe devices.

Lovelock can support extremely low latency networking because it does not incur a PCIe bus crossing between NIC and CPU (a special fabric is used instead). Current smart NIC hardware and drivers don't take advantage of this enough but can do so by directly writing to the internal CPU's cache line [44] or registers [26].

Memory on current smart NICs cannot support data-intensive workloads that rely on host memory for caching. We expect this limitation will disappear with CXL memory expansion (for in-memory caching) and swapping to far memory (for absorbing occasional surges).

Better isolation and performance predictability. Because they have less-capable CPUs, a Lovelock cluster can be efficiently utilized by a single application or a few applications of a single tenant. This setting eliminates cross-tenant interference and improves performance predictability. Avoiding host-level multi-tenant sharing also reduces the vulnerability of side channel attacks, thereby improving security isolation.

Scaling networking bandwidth. One of the main benefits of Lovelock is higher aggregate network bandwidth in configurations with $\phi > 1$. This can speed up applications (§5.2) but not those that can exploit fast intra-host communication to reduce inter-host traffic. Consider the all-reduce step in ML training. In a traditional cluster, all GPUs within a host reduce gradients over fast inter-GPU interconnect (e.g., NVLink) before reducing across hosts over slow datacenter network. If a Lovelock cluster scales by hosting fewer GPUs per smartNIC, the total datacenter network traffic for all-reduce operations will increase by ϕ.

Other data-intensive applications do not exhibit this behavior. Many data-intensive applications (e.g., Spark, BigQuery) are designed to use small-size worker nodes (4-16 cores per node/VM), and the number of worker nodes does not change, and neither would total network traffic, if these applications were hosted on a Lovelock cluster.

Scaling memory consumption. In Lovelock clusters with $\phi > 1$, the total memory consumed by application code and kernel will be higher than in a traditional cluster. We anticipate CXL-based memory disaggregation will alleviate this. A preliminary analysis of storage nodes shows that kernel's consumption is relatively small (1-2 GBs). We expect that memory used by applications will generally scale well since the input dataset is distributed, but this needs to be verified with additional analyses.

Networking and RPC performance. Smart NICs ASICs can offload packet processing and congestion control (§2.1), but likely not other networking components or RPC services. These may need to run on smart NIC CPUs, and we believe networking and RPC services can be optimized to run on these CPUs. For example, eRPC [27] demonstrates that a single core can achieve 10 million small RPCs per second or 75 Gbps with large messages. Our preliminary experiments with IPU E2000 suggest that a single ARM core can sustain over 25 Gbps with large message RPCs.

Data processing accelerators. Beyond network acceleration, smart NICs also have fixed-function data processing accelerators for crypto, compression, CRC, and copy. These can support infrastructure services like full-featured RPC, data center file systems, and logging without significantly taxing the smart NIC CPU and DRAM resources.

Network cost modeling. We can extend our model (§4) to reflect the increased cost of the network fabric when $\phi > 1$ by assuming that network cost scales linearly with cluster size (§5.2). However, this is pessimistic; smaller capacity increases might suffice to sustain application speedups, since applications will be slowed down by the smaller CPU. In addition, rack-local disaggregation can further reduce additional fabric capacity required. Right-sizing fabrics for applications will be crucial for Lovelock viability.

7 Related work

Offloading to smart NICs from host. Smart NICs are designed to offload various networking functions from host cores, and their effectiveness over general cores is proven [6, 7, 17]. Other research has explored the potential of offloading user applications beyond networking functions [30, 37], and demonstrated potential energy savings [31] and lower tail latency [15]. But, to our knowledge, no prior work has proposed replacing servers with smart NICs, as Lovelock does.

Disaggregated datacenter. Rapid improvements in network bandwidth and latency has enabled resource disaggregation [20, 38–40, 47]. For efficient disaggregation, prior work explores custom hardware and software for non-compute nodes (e.g., memory node) [23, 25, 41, 42]. Relative to Lovelock, these hardware-based disaggregation approaches require enormous software restructuring both in user applications and infrastructure applications.

Clustering Wimpy Storage Nodes. A fast array of wimpy nodes (FAWN) [9] demonstrated energy saving by clustering flash devices with wimpy embedded CPUs for large-scale key-value storage. They focused on improving energy efficiency for storage IO operations, whereas Lovelock targets data-intensive workloads with large data movement coupled with computation.

Acknowledgments

We thank Bob Felderman and Nandita Dukkipati for giving valuable feedback on earlier drafts.

References

[1] AMD Pensando Giglio data processing unit. https://www.amd.com/system/files/documents/pensando-giglio-product-brief.pdf.

[2] Intel Infrastructure Processing Unit (Intel IPU) ASIC E2000. https://www.intel.com/content/www/us/en/products/details/network-io/ipu/e2000-asic.html.

[3] Liquidio ii 10/25gbe adapter family. https://www.marvell.com/products/infrastructure-processors/liquidio-smart-nics/liquidio-ii-smart-nics.html.

[4] Netronome Agilio smartNICs. https://www.netronome.com/products/smartnic/overview/.

[5] Nvidia Bluefield-3 DPU datasheet. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-3-dpu.pdf.

[6] DPU POWER EFFICIENCY, White Paper, 2022. https://resources.nvidia.com/en-us-accelerated-networking-resource-library/nvidia-dpu-power-eff.

[7] ACCELERATING REDIS PERFORMANCE USING VMWARE VSPHERE 8 AND NVIDIA BLUEFIELD DPU, White Paper, 2023. https://resources.nvidia.com/en-us-accelerated-networking-resource-library/nvidia-vmware-redis.

[8] S. Agarwal, R. Agarwal, B. Montazeri, M. Moshref, K. Elmeleegy, L. Rizzo, M. A. de Kruijf, G. Kumar, S. Ratnasamy, D. Culler, and A. Vahdat. Understanding host interconnect congestion. In *Proceedings of the 21st ACM Workshop on Hot Topics in Networks*, HotNets '22, page 198–204, New York, NY, USA, 2022. Association for Computing Machinery.

[9] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan. Fawn: a fast array of wimpy nodes. In *Proceedings of the ACM SIGOPS 22nd Symposium on Operating Systems Principles*, SOSP '09, page 1–14, New York, NY, USA, 2009. Association for Computing Machinery.

[10] A. Audibert, Y. Chen, D. Graur, A. Klimovic, J. Simsa, and C. A. Thekkath. A case for disaggregation of ml data processing, 2022.

[11] D. Bansal, G. DeGrace, R. Tewari, M. Zygmunt, J. Grantham, S. Gai, M. Baldi, K. Doddapaneni, A. Selvarajan, A. Arumugam, B. Raman, A. Gupta, S. Jain, D. Jagasia, E. Langlais, P. Srivastava, R. Hazarika, N. Motwani, S. Tiwari, S. Grant, R. Chandra, and S. Kandula. Disaggregating stateful network functions. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 1469–1487, Boston, MA, Apr. 2023. USENIX Association.

[12] L. A. Barroso, U. Hölzle, and P. Ranganathan. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. 2019.

[13] S. Chen, C. Delimitrou, and J. F. Martínez. Parties: Qos-aware resource partitioning for multiple interactive services. In *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '19, page 107–120, New York, NY, USA, 2019. Association for Computing Machinery.

[14] W. Chiang, X. Liu, S. Si, Y. Li, S. Bengio, and C. Hsieh. Cluster-gcn: An efficient algorithm for training deep and large graph convolutional networks. *CoRR*, abs/1905.07953, 2019.

[15] S. Choi, S. J. Park, M. Shahbaz, B. Prabhakar, and M. Rosenblum. Toward scalable replication systems with predictable tails using programmable data planes. In *Proceedings of the 3rd Asia-Pacific Workshop on Networking 2019*, APNet '19, page 78–84, New York, NY, USA, 2019. Association for Computing Machinery.

[16] N. Du, Y. Huang, A. M. Dai, S. Tong, D. Lepikhin, Y. Xu, M. Krikun, Y. Zhou, A. W. Yu, O. Firat, B. Zoph, L. Fedus, M. Bosma, Z. Zhou, T. Wang, Y. E. Wang, K. Webster, M. Pellat, K. Robinson, K. Meier-Hellstern, T. Duke, L. Dixon, K. Zhang, Q. V. Le, Y. Wu, Z. Chen, and C. Cui. Glam: Efficient scaling of language models with mixture-of-experts, 2022.

[17] D. Firestone, A. Putnam, H. Angepat, D. Chiou, A. Caulfield, E. Chung, M. Humphrey, K. Ovtcharov, J. Padhye, D. Burger, D. Maltz, A. Greenberg, S. Mundkur, A. Dabagh, M. Andrewartha, V. Bhanu, H. K. Chandrappa, S. Chaturmohta, J. Lavier, N. Lam, F. Liu, G. Popuri, S. Raindel, T. Sapre, M. Shaw, G. Silva, M. Sivakumar, N. Srivastava, A. Verma, Q. Zuhair, D. Bansal, K. Vaid, and D. A. Maltz. Azure accelerated networking: Smartnics in the public cloud. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, April 2018.

[18] J. Fried, Z. Ruan, A. Ousterhout, and A. Belay. Caladan: Mitigating interference at microsecond timescales. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 281–297. USENIX Association, Nov. 2020.

[19] S. Gandhi and A. P. Iyer. P3: Distributed deep graph learning at scale. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, pages 551–568. USENIX Association, July 2021.

[20] P. X. Gao, A. Narayan, S. Karandikar, J. Carreira, S. Han, R. Agarwal, S. Ratnasamy, and S. Shenker. Network requirements for resource disaggregation. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 249–264, Savannah, GA, Nov. 2016. USENIX Association.

[21] A. Gonzalez, A. Kolli, S. Khan, S. Liu, V. Dadu, S. Karandikar, J. Chang, K. Asanovic, and P. Ranganathan. Profiling hyperscale big data processing. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23, New York, NY, USA, 2023. Association for Computing Machinery.

[22] D. Graur, D. Aymon, D. Kluser, T. Albrici, C. A. Thekkath, and A. Klimovic. Cachew: Machine learning input data processing as a service. In *2022 USENIX Annual Technical Conference (USENIX ATC 22)*, pages 689–706, Carlsbad, CA, July 2022. USENIX Association.

[23] Z. Guo, Y. Shan, X. Luo, Y. Huang, and Y. Zhang. Clio: A hardware-software co-designed disaggregated memory system. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '22, page 417–433, New York, NY, USA, 2022. Association for Computing Machinery.

[24] W. L. Hamilton, R. Ying, and J. Leskovec. Inductive representation learning on large graphs. *CoRR*, abs/1706.02216, 2017.

[25] C. Hu, C. Wang, S. Wang, N. Sun, Y. Bao, J. Zhao, S. Kashyap, P. Zuo, X. Chen, L. Xu, Q. Zhang, H. Feng, and Y. Shan. Skadi: Building a distributed runtime for data systems in disaggregated data centers. In *Proceedings of the 19th Workshop on Hot Topics in Operating Systems*, HOTOS '23, page 94–102, New York, NY, USA, 2023. Association for Computing Machinery.

[26] S. Ibanez, A. Mallery, S. Arslan, T. Jepsen, M. Shahbaz, C. Kim, and N. McKeown. The nanopu: A nanosecond network stack for datacenters. In *15th USENIX Symposium on Operating Systems Design and Implementation (OSDI 21)*, pages 239–256. USENIX Association, July 2021.

[27] A. Kalia, M. Kaminsky, and D. Andersen. Datacenter RPCs can be general and fast. In *16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19)*, pages 1–16, Boston, MA, Feb. 2019. USENIX Association.

[28] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[29] Z. Lin, C. Li, Y. Miao, Y. Liu, and Y. Xu. Pagraph: Scaling GNN training on large graphs via computation-aware caching. In R. Fonseca, C. Delimitrou, and B. C. Ooi, editors, *SoCC '20: ACM Symposium on Cloud Computing, Virtual Event, USA, October 19-21, 2020*, pages 401–415. ACM, 2020.

[30] M. Liu, T. Cui, H. Schuh, A. Krishnamurthy, S. Peter, and K. Gupta. Offloading distributed applications onto smartnics using ipipe. In *Proceedings of the ACM Special Interest Group on Data Communication*, SIGCOMM '19, page 318–333, New York, NY, USA, 2019. Association for Computing Machinery.

[31] M. Liu, S. Peter, A. Krishnamurthy, and P. M. Phothilimthana. E3: Energy-Efficient microservices on SmartNIC-Accelerated servers. In *2019 USENIX Annual Technical Conference (USENIX ATC 19)*, pages 363–378, Renton, WA, July 2019. USENIX Association.

[32] T. Liu, Y. Chen, D. Li, C. Wu, Y. Zhu, J. He, Y. Peng, H. Chen, H. Chen, and C. Guo. BGL: GPU-Efficient GNN training by optimizing graph data I/O and preprocessing. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 103–118, Boston, MA, Apr. 2023. USENIX Association.

[33] D. Lo, L. Cheng, R. Govindaraju, P. Ranganathan, and C. Kozyrakis. Heracles: Improving resource efficiency at scale. In *Proceedings of the 42nd Annual International Symposium on Computer Architecture*, ISCA '15, page 450–462, New York, NY, USA, 2015. Association for Computing Machinery.

[34] D. Narayanan, A. Harlap, A. Phanishayee, V. Seshadri, N. R. Devanur, G. R. Ganger, P. B. Gibbons, and M. Zaharia. Pipedream: Generalized pipeline parallelism for dnn training. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*, SOSP '19, page 1–15, New York, NY, USA, 2019. Association for Computing Machinery.

[35] R. Neugebauer, G. Antichi, J. F. Zazo, Y. Audzevich, S. López-Buedo, and A. W. Moore. Understanding pcie performance for end host networking. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, SIGCOMM '18, page 327–341, New York, NY, USA, 2018. Association for Computing Machinery.

[36] S. J. Park, J. Fried, S. Kim, M. Alizadeh, and A. Belay. Efficient strong scaling through burst parallel training. In D. Marculescu, Y. Chi, and C. Wu, editors, *Proceedings of Machine Learning and Systems*, volume 4, pages 748–761, 2022.

[37] P. M. Phothilimthana, M. Liu, A. Kaufmann, S. Peter, R. Bodik, and T. Anderson. Floem: A programming system for NIC-Accelerated network applications. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 663–679, Carlsbad, CA, Oct. 2018. USENIX Association.

[38] Z. Ruan, S. Li, K. Fan, M. K. Aguilera, A. Belay, S. J. Park, and M. Schwarzkopf. Unleashing true utility computing with quicksand. In *Proceedings of the 19th Workshop on Hot Topics in Operating Systems*, HOTOS '23, page 196–205, New York, NY, USA, 2023. Association for Computing Machinery.

[39] Z. Ruan, S. J. Park, M. K. Aguilera, A. Belay, and M. Schwarzkopf. Nu: Achieving Microsecond-Scale resource fungibility with logical processes. In *20th USENIX Symposium on Networked Systems Design and Implementation (NSDI 23)*, pages 1409–1427, Boston, MA, Apr. 2023. USENIX Association.

[40] Z. Ruan, M. Schwarzkopf, M. K. Aguilera, and A. Belay. AIFM: High-Performance, Application-Integrated far memory. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 315–332. USENIX Association, Nov. 2020.

[41] Y. Shan, Y. Huang, Y. Chen, and Y. Zhang. LegoOS: A disseminated, distributed OS for hardware resource disaggregation. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, pages 69–87, Carlsbad, CA, Oct. 2018. USENIX Association.

[42] Y. Shan, W. Lin, Z. Guo, and Y. Zhang. Towards a fully disaggregated and programmable data center. In *Proceedings of the 13th ACM SIGOPS Asia-Pacific Workshop on Systems*, APSys '22, page 18–28, New York, NY, USA, 2022. Association for Computing Machinery.

[43] N. Sundar, B. Burres, Y. Li, D. Minturn, B. Johnson, and N. Jain. An in-depth look at the Intel IPU E2000. In *Proceedings of the IEEE International Solid-State Circuits Conference*, San Francisco, CA, Feb. 2023.

[44] M. Sutherland, S. Gupta, B. Falsafi, V. Marathe, D. Pnevmatikatos, and A. Daglis. The nebula rpc-optimized architecture. In *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, pages 199–212, 2020.

[45] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Liò, and Y. Bengio. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

[46] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes. Large-scale cluster management at google with borg. In *Proceedings of the Tenth European Conference on Computer Systems*, EuroSys '15, New York, NY, USA, 2015. Association for Computing Machinery.

[47] Y. Zhou, H. M. G. Wassel, S. Liu, J. Gao, J. Mickens, M. Yu, C. Kennelly, P. Turner, D. E. Culler, H. M. Levy, and A. Vahdat. Carbink: Fault-Tolerant far memory. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pages 55–71, Carlsbad, CA, July 2022. USENIX Association.