The Case for In-Memory OLAP on "Wimpy" Nodes

Andrew Crotty, Alex Galakatos, Connor Luckett, Ugur Cetintemel
Department of Computer Science, Brown University
firstname_lastname@brown.edu

***Abstract*—Research projects will often use the latest hardware to achieve orders-of-magnitude performance improvements while ignoring the (usually hefty) associated price tag. Real-world deployments typically follow suit, requiring expensive computing infrastructures that cost even more to power and cool.**

In this paper, we challenge the conventional wisdom that high-end hardware is absolutely necessary for state-of-the-art performance and instead advocate for a radically different approach based on cheap *single-board computers* (SBCs). While others have previously explored similar ideas for computationally simple and easily partitionable use cases (e.g., key-value stores), so-called "wimpy" nodes have traditionally been rejected as unsuitable for more complex workloads. We believe, however, that recent hardware advancements driven by the mobile computing market call this orthodoxy into question. For example, our microbenchmarks show that one popular SBC, the Raspberry Pi 3B+, offers single-core compute performance that is surprisingly competitive with many server-grade Intel Xeon and ARM-based CPUs at a fraction of the cost and energy consumption.

To make our case, we conducted an extensive experimental study, beginning with a series of microbenchmarks to identify the strengths and weaknesses of SBCs relative to server-grade CPUs. Then, to evaluate the ability of SBCs to handle more complex use cases, we analyzed the performance of an in-memory OLAP workload in both single-node and distributed settings. Overall, our results demonstrate up to several orders of magnitude in cost reductions coupled with substantial energy savings when compared to traditional on-premises and cloud deployments, all without a significant increase in absolute runtimes.

I. Introduction

Modern data analytics frameworks traditionally favor high-end hardware, with deployments typically ranging from a single machine to small clusters [1], [2]. For example, Amazon EC2 [3] offers the X1 instance type, which is optimized for "in-memory databases and big data processing engines." These X1 instances include up to four Intel Xeon E7-8880 v3 CPUs [4]—each costing almost $6,000—as well as up to 2 TB of memory and nearly 4 TB of local SSD storage. Other cloud providers, including Google Cloud [5] and Microsoft Azure [6], have similar configurations available.

On the other end of the spectrum are inexpensive *single-board computers* (SBCs), which integrate all hardware components necessary for a full-fledged computer into a single circuit board. For example, the SBC shown in Figure 1 is the Raspberry Pi 3B+ [7], which costs only $35 despite boasting a quad-core processor, 1 GB of memory, and a variety of ports for peripheral devices (e.g., USB, microSD, HDMI). In recent years, SBCs have become increasingly commonplace for certain data management use cases, most notably sensor networks and edge processing [8], [9], [10], [11], [12].

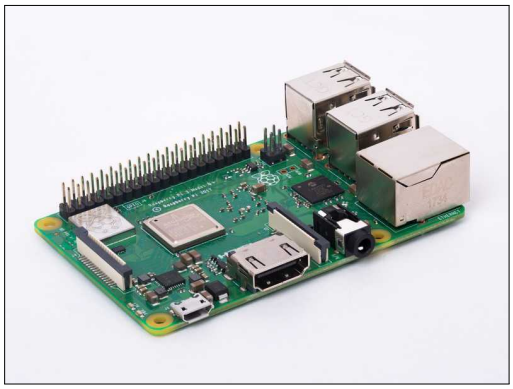


Fig. 1: Raspberry Pi 3B+

At the same time, the ubiquity of mobile devices has led to staggering hardware advancements that have also benefited SBCs, since they share many of the same components [13], [14]. While older Raspberry Pi models, for instance, had only weak single-core CPUs, current versions include powerful multi-core CPUs with advanced features (e.g., superscalar processing, out-of-order execution). Surprisingly, our microbenchmarks even show that the single-core compute performance of a Raspberry Pi 3B+ is competitive with many server-grade Intel Xeon and ARM-based CPUs.

SBCs also consume substantially less energy than traditional servers. For example, at peak CPU utilization, a Raspberry Pi 3B+ draws at most 5.1 watts, whereas server-grade CPUs typically draw hundreds of watts under normal load [14]. This estimate does not even include other essential components (e.g., memory, persistent storage), which can often represent the majority of a server's total power draw [15], [16], [17], [18], nor does it consider the operational electricity costs, which typically eclipse the cost of the initial hardware purchase [19]. Moreover, as a consequence of their low energy requirements, SBCs produce significantly less heat, thereby eliminating the need for special cooling equipment.

Based on these observations, we believe that SBCs have actually become a viable alternative to traditional servers. Unlike past work that focuses on computationally simple and easily parallelizable use cases (e.g., key-value stores [20], web servers [21], search engines [22]), we demonstrate that SBCs are indeed capable of handling complex workloads like in-memory OLAP, and our results open the door for exploring the feasibility of running other computationally intensive workloads (e.g., machine learning).

2375-026X/21/$31.00 ©2021 IEEE
DOI 10.1109/ICDE51399.2021.00069

This paper makes the case that SBCs can provide significantly more "bang for the buck" in the context of these more complex workloads, especially for use cases that are not latency-sensitive (e.g., batch processing). Through an extensive experimental study, we show that SBCs can achieve competitive in-memory OLAP performance while simultaneously offering two major advantages over traditional servers: (1) lower cost and (2) reduced energy consumption. In fact, our results directly contradict past studies [23], [21] that dismiss SBCs as unsuitable for computationally intensive workloads.

In summary, we make the following contributions:

- Using a series of microbenchmarks, we identify the strengths and weaknesses of a popular SBC, the Raspberry Pi 3B+, relative to a wide range of server-grade Intel Xeon and ARM-based CPUs.
- We conduct TPC-H benchmarks for all comparison points, evaluating the Raspberry Pi 3B+ in both a single-node and distributed setting. As part of our study, we also built WIMPI, a cluster of Raspberry Pi 3B+ nodes that allows us to test different cluster sizes.
- Based on these results, we provide a detailed cost and energy consumption analysis, and we show that a cluster of SBCs can offer substantial benefits in both dimensions while achieving competitive absolute runtimes.

The remainder of this paper is organized as follows. Section II presents the setup and results of our experimental study. Then, in Section III, we analyze these results in terms of both cost and energy consumption, as well as highlight other important considerations for complex workloads on SBCs. Finally, we discuss related work in Section IV and summarize our findings in Section V.

II. EXPERIMENTAL STUDY

This section presents our experimental study, which compares a popular SBC, the Raspberry Pi 3B+, to a variety of server-grade Intel Xeon and ARM-based CPUs. We first provide an overview of the tested hardware and describe our prototype WIMPI cluster, followed by a detailed discussion of the results obtained from our benchmarks.

A. Tested Hardware

We begin with a brief overview of the hardware specifications for each of the comparison points in our study, which are summarized in Table I. For simplicity, we focus only on the characteristics of the CPUs, since they have the largest impact on in-memory OLAP performance. Other necessary hardware components (e.g., persistent storage) can vary greatly and have virtually no performance implications for our target use case, so we do not consider them in this study.

1) On-Premises: Our research group operates two traditional servers, denoted in the table as the On-Premises group. The first server (`op-e5`) has an Intel Xeon E5-2660 v2 CPU [24], which is on the lower end of our comparison points, while the second (`op-gold`) has a more recently released Intel Xeon Gold 6150 CPU [25].

Since these two CPUs were developed for the retail market, Intel makes detailed product specifications publicly available. In particular, we focus on the manufacturer's suggested retail price (MSRP), which is the list price recommended by Intel for retail sellers, and the thermal design power (TDP), which refers to the average power dissipation for CPUs operating at base frequency under a heavy workload. We use this information for the cost and energy analyses presented in Section III. Other important CPU characteristics include clock frequency, number of cores, and size of the last level cache (LLC), all of which can have a substantial impact on performance for an in-memory OLAP workload.

Note that we chose not to include estimated hourly operating costs for the On-Premises group, which we could calculate as the total power consumption of all hardware components multiplied by the kilowatt-hour energy price. We felt that this metric would unfairly disadvantage these servers, since they were built for general-purpose use prior to our study and do not necessarily have the most energy-efficient components. For this same reason, our energy comparisons (Section III-B) for the servers conservatively use only the TDP of the CPU.

2) Cloud: To expand the range of tested hardware, we also included seven different instance types from Amazon EC2, labeled as the Cloud group in the table. Five instance types had Intel Xeon CPUs, with the first three (`c4.8xlarge`, `m4.10xlarge`, `m4.16xlarge`) from the E5 family and the other two (`z1d.metal`, `m5.metal`) from the newer Platinum family. The remaining two instance types (`a1.metal`, `c6g.metal`) had custom ARM-based CPUs built by AWS. Specifically, the `a1.metal` instance uses the first-generation Graviton, which has four 2.3 GHz quad-core ARM Cortex-A72 CPUs, whereas the `c6g.metal` instance has the newer Graviton2, which is based on the ARM Neoverse N1 microarchitecture and has a total of 64 cores on a single socket.

Although Amazon EC2 offers many options, we chose these specific instance types because they are guaranteed (or highly likely) to be backed by a single physical processor rather than sharing resources with other users in a virtualized environment, thereby enabling more accurate performance measurements. However, since all of these CPUs are custom SKUs for Amazon EC2, the MSRP and TDP are not publicly available. Therefore, for the Cloud group, we include only the hourly price [26] charged by Amazon EC2, which ranges from $0.408 for the `a1.metal` instance to $4.608 for the `m5.metal` instance.

3) SBC: The Raspberry Pi is one of the most popular series of SBCs, with use cases ranging from small hobby projects to industrial IoT applications. Due to the lightweight requirements for most of these use cases, Raspberry Pis were designed to have a small form factor and power-conscious hardware at a very low price point. These characteristics, though, might give the false impression that currently available Raspberry Pi models share the same weak performance of wimpy nodes explored in previous studies. On the contrary, SBCs have recently benefited greatly from rapid hardware advancements in the mobile device market, and we believe

Category	Name	CPU	Frequency	Cores	LLC	MSRP	Hourly	TDP
On-Premises	op-e5	Intel Xeon E5-2660 v2	2.2 GHz	10	25 MB	$1,389	-	95 W
	op-gold	Intel Xeon Gold 6150	2.7 GHz	18	24.75 MB	$3,358	-	165 W
Cloud	c4.8xlarge	Intel Xeon E5-2666 v3	2.9 GHz	9	25 MB	-	$1.591	-
	m4.10xlarge	Intel Xeon E5-2676 v3	2.4 GHz	10	30 MB	-	$2.00	-
	m4.16xlarge	Intel Xeon E5-2686 v4	2.3 GHz	16	45 MB	-	$3.20	-
	z1d.metal	Intel Xeon Platinum 8151	3.4 GHz	12	24.75 MB	-	$4.464	-
	m5.metal	Intel Xeon Platinum 8259CL	2.5 GHz	24	35.75 MB	-	$4.608	-
	a1.metal	AWS Graviton	2.3 GHz	16	8 MB	-	$0.408	-
	c6g.metal	AWS Graviton2	2.5 GHz	64	32 MB	-	$2.176	-
SBC	Pi 3B+	ARM Cortex-A53	1.4 GHz	4	512 KB	$35	$0.0004	5.1 W

TABLE I: Hardware Specifications

that they have become competitive with traditional servers in the context of more complex workloads like in-memory OLAP.

Specifically, we selected the Raspberry Pi 3B+ to represent the SBC category in our study. The Raspberry Pi 3B+ has a 1.4 GHz quad-core ARM Cortex-A53 CPU with a 512 KB LLC. The Cortex-A53 has found widespread use in SBCs as well as a variety of other devices, including Amazon Fire tablets, Roku streaming media players, and the Nintendo Switch video game console.

The current MSRP for the Raspberry Pi 3B+ is just $35, although the price is likely to decrease as newer models (e.g., the Raspberry Pi 4B [27]) come to market. For the TDP shown in Table I, we actually report the maximum power draw of the entire SBC (i.e., 5.1 watts), not the TDP of the CPU alone. We then estimate the hourly operating cost for a single Raspberry Pi 3B+ by multiplying the maximum possible power draw by the US national average price per kilowatt-hour [28], which yields a rate of less than $0.0004 per hour.

These choices actually result in a pessimistic analysis for the Raspberry Pi 3B+, since they do not always operate at sustained maximum power draw. Moreover, as mentioned, our analyses for the traditional servers do not include the energy consumption of hardware components other than the CPU. We further explain the rationale behind these decisions in Section III.

B. WimPi Cluster

In spite of its low price point, the Raspberry Pi 3B+ clearly has quite impressive specifications, but it also has certain limitations that might prevent a single node from effectively handling larger data sizes. For example, the Raspberry Pi 3B+ has only four physical cores, whereas the server-grade CPUs that we tested range from nine physical cores (18 virtual cores when considering Hyper-Threading) to as many as 64. We therefore anticipate that many use cases will actually require a cluster comprised of several Raspberry Pi 3B+ nodes, so one of the key questions we sought to answer in our study is how many nodes would be required to match (or exceed) the performance of a traditional server.

To answer this question, we built a prototype Raspberry Pi 3B+ cluster, called WIMPI, that allows us to evaluate query performance with different cluster sizes. WIMPI consists of 24 Raspberry Pi 3B+ nodes connected via Ethernet on a Gigabit switch, offering a total of 96 physical cores and 24 GB of aggregate memory. The cluster requires four power supplies, each with six ports that connect to the nodes via USB to Micro-USB cables. The total cost was approximately $840 (i.e., 24 nodes at $35 each), excluding peripherals (e.g., microSD cards, Ethernet cables). However, depending on quality and bulk purchasing discounts, these peripherals only increase the cost of each node by roughly $10–15. Section III-A provides a more detailed price analysis.

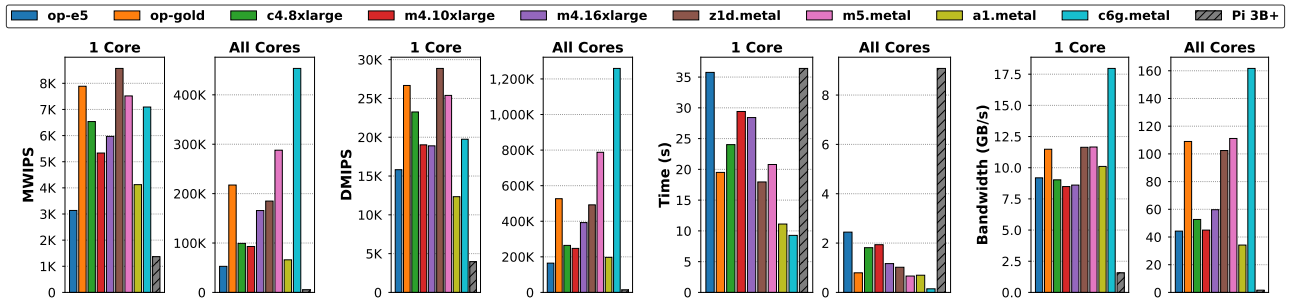
In addition to performance and cost, energy consumption is another major factor for real-world deployments. As a back-of-the-envelope calculation using the peak power consumption of 5.1 watts for each Raspberry Pi 3B+ node, the total maximum power draw of our entire WIMPI cluster is roughly 122 watts (i.e., 24 nodes at 5.1 watts each). On the other hand, the power consumption for a single CPU in the On-Premises servers ranges from 95–165 watts, as shown in Table I.

Decreased power consumption not only lowers operational electricity costs but also has the added benefit of reducing the associated cooling costs, which can be substantial [19], [29]. In fact, given the dramatically lower power consumption, our WIMPI cluster requires none of the expensive cooling equipment necessary for operating traditional servers.

C. Microbenchmarks

As a first step, we obtained a rough performance baseline for each comparison point using several common microbenchmark suites, many of which appear in similar previous studies [14], [21]. In particular, we focus on CPU performance and memory bandwidth, which represent the two primary bottlenecks for in-memory OLAP workloads. Additionally, since we consider the Raspberry Pi 3B+ in a distributed setting, we also measured the network bandwidth between nodes in our WIMPI cluster.

1) CPU: We evaluated CPU performance using three well-known microbenchmarks: (1) Whetstone [30], (2) Dhrystone [31], and (3) `sysbench` [32]. On every comparison point, we ran each microbenchmark first using only a single core and then with all available cores. For the Intel Xeon CPUs in the latter set of experiments, we set the number of threads to twice the count of physical cores shown in Table I, as we found that Hyper-Threading improved performance. On the other hand, the Raspberry Pi 3B+ and ARM-based Cloud servers (i.e., `a1.metal` and `c6g.metal`) used only one thread per physical core.



(a) Whetstone **(b) Dhrystone** **(c) Prime Testing** **(d) Memory Bandwidth**

Fig. 2: Microbenchmark Results

Figures 2a and 2b plot the MWIPS (Millions of Whetstone Instructions Per Second) and DMIPS (Dhrystone Millions of Instructions Per Second), respectively. Higher values (i.e., more instructions per second) are better.

Both of these microbenchmarks show that the single-core compute performance of the Raspberry Pi 3B+ is only between 2–3× worse than the `op-e5` server, ranging up to roughly 5–6× worse than the higher-end `op-gold` and `m5.metal` servers. This result is especially impressive, considering that the Raspberry Pi 3B+ costs roughly two orders of magnitude less than these server-grade CPUs. The `z1d.metal` instance, which has a sustained clock speed of 3.4 GHz, unsurprisingly exhibits the best single-core performance.

When looking at the results for all cores, though, the server-grade CPUs range from 10–90× more powerful, due to their significantly greater parallelism. In particular, the AWS Graviton2 CPU in the `c6g.metal` instance type, which has 64 physical cores, achieves the best performance by a wide margin. For comparison, recall that the Raspberry Pi 3B+ has only four cores.

The results for the `sysbench` [32] microbenchmark, which involves a tight loop testing for prime numbers, are shown in Figure 2c. Unlike the previous two microbenchmarks, lower values (i.e., shorter runtimes) are better.

Surprisingly, the single-core performance of a Raspberry Pi 3B+ is nearly identical to the Intel E5-2660 v2 in the `op-e5` server. In fact, the other server-grade CPUs only offer between 1.2–3.9× better performance. When considering the performance for all cores, the server-grade CPUs only beat the Raspberry Pi 3B+ by 4–14×, with the exception of the `c6g.metal` instance type.

Overall, these microbenchmark results suggest that the Raspberry Pi 3B+ should exhibit unexpectedly good performance for computationally intensive use cases, which directly contradicts the conclusions of another recent study [21]. Specifically, the authors found that an Intel Edison SBC could only achieve roughly 5–6% of the single-core performance of a server-grade Xeon CPU. When examining the performance for all cores, they observed a roughly 100× slowdown, leading them to the conclusion that SBCs were ill-suited for CPU-bound workloads.

2) Memory: Another critical aspect to consider is the memory bandwidth of the CPU. Whereas memory access latency might be more important for key-value stores or OLTP applications that perform many point lookups, memory bandwidth can often become a bottleneck in OLAP workloads, which are typified by scanning large tables.

We use the memory bandwidth test included in the `sysbench` toolkit, which allocates a large buffer of memory and measures the time required for the CPU to sequentially read the entire buffer. Again, we run the microbenchmarks on each comparison point using both a single core and all available cores. Unlike the CPU microbenchmarks, we found that Hyper-Threading did not improve memory bandwidth results, so we only used one thread per physical core for the Intel Xeon CPUs.

As shown in Figure 2d, we see that a single core on the Raspberry Pi 3B+ has roughly 5–11× lower memory bandwidth than the server-grade CPUs. When looking at all cores, the bandwidth for the Raspberry Pi 3B+ remains nearly the same, since it has a single memory channel that can be almost fully saturated by one core. The server-grade CPUs therefore exhibit between 20–99× higher bandwidth than the Raspberry Pi 3B+. These results comport with another recent study that investigated similar hardware [21].

Although our 24-node WIMPI cluster provides aggregate memory bandwidth equivalent to the `op-e5` and `m4.10xlarge` CPUs, we would need to triple the number of nodes to match the `op-gold` and `m5.metal` CPUs. Even then, the total cost for the entire cluster would only be $2,500, which is still less expensive than a single Xeon Gold 6150 CPU. When adding the cost of all the other necessary server components, even such a large Raspberry Pi 3B+ cluster still maintains a large price advantage.

3) Network: Network bandwidth is not relevant for the traditional servers in our study, since all processing occurs on a single node. However, since we consider the Raspberry Pi 3B+ in a distributed setting, network bandwidth will become an increasingly important consideration as the cluster size grows.

Again, nodes in our WIMPI cluster are connected via Ethernet on a Gigabit switch, but each node can only use about 20% of the available bandwidth because the Ethernet

Category	Name	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17	Q18	Q19	Q20	Q21	Q22
On-Premises	op-e5	0.161	0.008	0.080	0.061	0.082	0.028	0.052	0.116	0.116	0.062	0.017	0.036	0.196	0.019	0.034	0.156	0.101	0.130	0.027	0.045	0.155	0.112
	op-gold	0.056	0.008	0.046	0.025	0.041	0.012	0.024	0.069	0.055	0.031	0.011	0.020	0.121	0.011	0.015	0.084	0.051	0.063	0.020	0.022	0.199	0.063
Cloud	c4.8xlarge	0.054	0.008	0.021	0.016	0.020	0.006	0.022	0.037	0.033	0.017	0.006	0.011	0.097	0.006	0.011	0.045	0.022	0.050	0.018	0.016	0.068	0.038
	m4.10xlarge	0.056	0.007	0.021	0.017	0.021	0.007	0.021	0.041	0.034	0.019	0.006	0.013	0.111	0.007	0.012	0.048	0.022	0.057	0.021	0.018	0.087	0.044
	m4.16xlarge	0.043	0.007	0.023	0.015	0.021	0.006	0.023	0.043	0.032	0.022	0.006	0.014	0.116	0.009	0.012	0.045	0.016	0.059	0.029	0.020	0.237	0.043
	z1d.metal	0.073	0.012	0.079	0.052	0.057	0.027	0.035	0.096	0.083	0.054	0.024	0.032	0.196	0.018	0.031	0.167	0.089	0.084	0.037	0.047	0.169	0.094
	m5.metal	0.034	0.010	0.033	0.023	0.026	0.008	0.025	0.053	0.043	0.031	0.010	0.018	0.135	0.011	0.017	0.074	0.027	0.064	0.031	0.024	0.248	0.064
	a1.metal	0.270	0.009	0.062	0.064	0.087	0.025	0.071	0.126	0.123	0.053	0.018	0.046	0.330	0.015	0.026	0.190	0.077	0.135	0.024	0.032	0.085	0.143
	c6g.metal	0.049	0.005	0.045	0.026	0.047	0.011	0.038	0.079	0.057	0.052	0.011	0.032	0.204	0.020	0.018	0.117	0.040	0.083	0.017	0.022	0.620	0.081
SBC	Pi 3B+	1.772	0.044	0.227	0.222	0.283	0.099	0.486	0.244	0.684	0.221	0.034	0.154	1.771	0.076	0.093	0.302	0.220	0.394	0.140	0.141	0.603	0.269

TABLE II: Runtimes (s) for SF 1

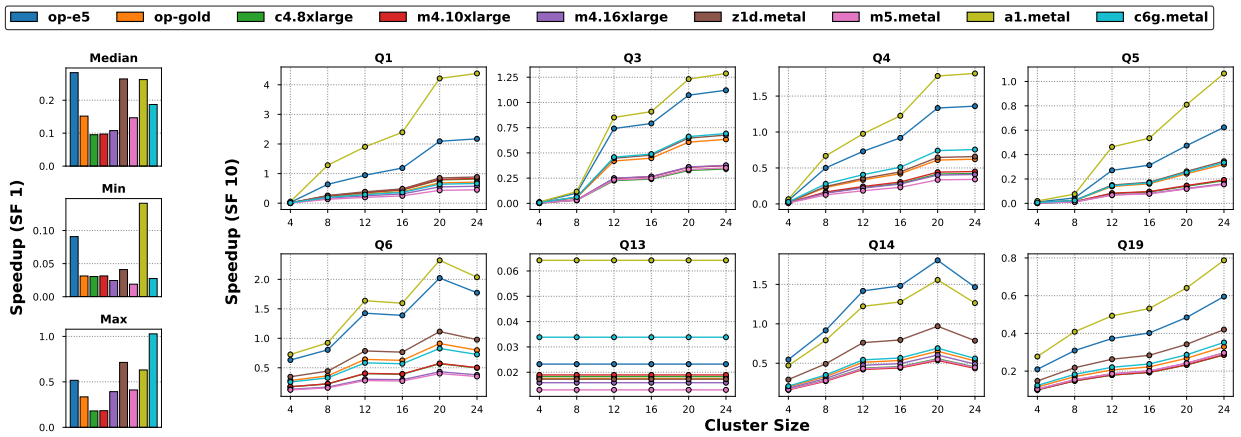


Fig. 3: Speedups for SF 1 (left) and SF 10 (right)

Category	Name	Q1	Q3	Q4	Q5	Q6	Q13	Q14	Q19
On-Premises	op-e5	1.474	0.603	0.465	0.542	0.191	2.405	0.153	0.131
	op-gold	0.482	0.341	0.212	0.278	0.086	1.817	0.055	0.072
Cloud	c4.8xlarge	0.554	0.183	0.144	0.161	0.054	1.897	0.047	0.063
	m4.10xlarge	0.566	0.201	0.154	0.167	0.054	1.963	0.045	0.063
	m4.16xlarge	0.388	0.203	0.140	0.140	0.041	1.644	0.051	0.065
	z1d.metal	0.600	0.364	0.225	0.300	0.105	1.787	0.082	0.092
	m5.metal	0.306	0.189	0.117	0.135	0.038	1.351	0.047	0.065
	a1.metal	2.972	0.692	0.620	0.925	0.219	6.651	0.132	0.173
	c6g.metal	0.452	0.372	0.258	0.290	0.078	3.505	0.059	0.077
SBC	Pi 3B+ ×4	57.814	53.424	9.492	47.147	0.303	103.604	0.280	0.624
	Pi 3B+ ×8	2.319	5.920	0.928	12.165	0.238	103.604	0.167	0.423
	Pi 3B+ ×12	1.561	0.813	0.636	1.999	0.134	103.604	0.108	0.351
	Pi 3B+ ×16	1.242	0.761	0.506	1.730	0.138	103.604	0.103	0.325
	Pi 3B+ ×20	0.705	0.562	0.348	1.143	0.094	103.604	0.085	0.270
	Pi 3B+ ×24	0.678	0.538	0.342	0.868	0.108	103.604	0.104	0.220

TABLE III: Runtimes (s) for SF 10

port shares a bus with the lower-bandwidth USB 2.0 ports. To validate this estimate, we measured the network bandwidth between two WIMPI nodes using the `iperf` [33] tool. The result was a transfer speed of about 220 Mbps, which is in line with the expected bandwidth limitation.

D. TPC-H

Unlike previous work, the goal of this paper is to make the case that so-called "wimpy" nodes can actually handle more complex workloads like in-memory OLAP. We first ran the well-known TPC-H benchmark on each comparison point at scale factors (SFs) 1 and 10 using a popular OLAP DBMS. Then, to identify any hidden performance differences that may have been obscured by the system-level benchmarks, we also conducted a series of low-level experiments that isolated and evaluated three different query execution strategies. In the following, we describe the results of our TPC-H evaluation.

1) SF 1: We began with the relatively small (but nontrivial) SF 1, which fits entirely in memory on all comparison points, including the 1 GB of available memory on a single Raspberry Pi 3B+. To run the benchmark, we chose MonetDB [34], a mature and widely used column-oriented DBMS with many advanced features. We used `gcc-9.3.0` to build MonetDB v11.39.11 (Oct2020 Release) from source on each comparison point for consistency, since no compatible pre-built release binary existed for the ARM-based comparison points.

The absolute runtimes appear in Table II, with speedups shown in Figure 3. For almost all queries, the Raspberry Pi 3B+ achieves reasonable absolute runtimes and is, on average, only about 10× slower than the traditional servers. More specifically, the median performance of the Raspberry Pi 3B+ relative to the servers ranges from about 0.1–0.3×, which is impressive considering that the much more expensive server-grade CPUs have higher clock speeds, more cores, larger LLCs, and significantly greater memory bandwidths. In some cases (e.g., Q11), the relative performance of the Raspberry Pi 3B+ is as high as 0.5–0.7× of the traditional servers, and in one case (Q21), the Raspberry Pi 3B+ is even able to outperform the `c6g.metal` instance.

As expected, the Raspberry Pi 3B+ is most competitive for CPU-bound queries. Specifically, we observed the smallest gaps in performance for queries that either do not include the very large `lineitem` table (e.g., Q11, Q16) or have highly

selective predicates (e.g., Q8). On the other hand, we observed the worst performance for Q1, which scans almost the entire `lineitem` table and is therefore heavily memory-bound on the Raspberry Pi 3B+.

2) SF 10: Next, we increased the database size to SF 10, which required us to utilize multiple Raspberry Pi 3B+ nodes in the WIMPI cluster to keep all processing purely in memory. Consequently, we also evaluated the distributed scalability of WIMPI by running queries at six different cluster sizes, ranging from 4–24 nodes.

We again ran the benchmarks with MonetDB, although we did not use the built-in distributed query processing capabilities due to extremely poor performance for queries that produce large intermediate results (e.g., during join processing). Instead, we used the MonetDB Python client API to build a simple driver program that aggregates partial results from each node in the cluster. Moreover, since MonetDB does not currently provide transparent partitioning of large tables, we needed to manually distribute the data across the nodes in the WIMPI cluster. Therefore, similar to the setup in previous experimental studies of wimpy clusters [35], [36], we fully replicated all tables except `lineitem`, which we partitioned evenly across each of the nodes on the `l_orderkey` column. Section III-C provides further details about this setup.

Due to space constraints, we focus only on the subset of TPC-H queries used in recent papers [37], [38] that cover the main chokepoints of the TPC-H benchmark [39]. The absolute runtimes appear in Table III, with speedups again shown in Figure 3. Overall, the Raspberry Pi 3B+ was much more competitive in the distributed SF 10 experiments compared to the single-node setting for SF 1. With larger cluster sizes, WIMPI can often achieve greater than 0.5× the performance of the traditional servers, and in five of the eight tested queries (i.e., Q1, Q3, Q4, Q6, Q14), it can even outperform at least one of the comparison points.

For several queries (e.g., Q1, Q3, Q4, Q5), WIMPI performance did not scale uniformly as we added more nodes. Rather, we observed extremely poor performance at the initial cluster size of four nodes, followed by a huge jump (as much as 10–100× performance improvement) after doubling or tripling the number of nodes. For example, recall the poor Raspberry Pi 3B+ performance on Q1 from the previous SF 1 experiments. Since Q1 is heavily memory-bound, adding more nodes to the cluster increases the aggregate memory bandwidth and alleviates the bottleneck. At some point, the data may be able to fit entirely in LLC, avoiding the memory bandwidth bottleneck altogether.

In some cases (e.g., Q6, Q14), however, we noticed that increasing the cluster size beyond a certain point had diminishing returns, since network latency becomes the bottleneck. Similarly, notice that adding more nodes has no impact on the performance of Q13. Since this query does not include the partitioned `lineitem` table, it is executed using only a single node. A more sophisticated distributed query processing approach that could also parallelize joins between other tables would likely yield performance trends similar to those observed for the other queries, but this type of optimization is beyond the scope of this paper.

3) Execution Strategies: While benchmarking a real system like MonetDB gives important insights into how SBCs like the Raspberry Pi 3B+ might perform in practice, evaluating performance at such a high level introduces many confounding factors that could obscure the impacts of hardware characteristics. Therefore, to supplement our system-level benchmarks, we also conducted a low-level analysis using three different query execution paradigms that appear in a recent paper [38], including: (1) data-centric, (2) hybrid, and (3) access-aware. Each strategy was hand-coded in C and compiled with `gcc-9.3.0`, with all experiments run single-threaded.

We again use the same eight representative TPC-H queries from the distributed experiments, returning to SF 1 in order to ensure that the data fits in the memory of a single Raspberry Pi 3B+. Due to space constraints, we only compare the performance of the Raspberry Pi 3B+ to the On-Premises servers, since the Cloud servers exhibited similar trends. Figure 4 shows the results.

The runtimes for the Raspberry Pi 3B+ range between 2–19× slower than the same strategy executed on the traditional servers, comporting with the previous results for MonetDB at SF 1. The median performance gap, though, is now significantly reduced, due to the elimination of system-level overheads. As seen in the original paper [38], access-aware always performs the best and data-centric the worst, with hybrid somewhere in between.

Surprisingly, all of the strategies exhibit fairly consistent behavior across comparison points, although the performance advantages of the hybrid and access-aware strategies on the Raspberry Pi 3B+ were less pronounced. Given the limited memory bandwidth of the Raspberry Pi 3B+, this result is somewhat unexpected for the access-aware strategy in particular, which often trades extra memory accesses in exchange for more consistent access patterns.

III. DISCUSSION

As mentioned, the two main advantages of the Raspberry Pi 3B+ compared to traditional servers are (1) much lower cost and (2) significantly reduced energy consumption. In this section, we evaluate the TPC-H performance results obtained from our study (Section II-D) relative to these two metrics.

Specifically, we normalize the absolute runtimes for each comparison point by the metric under consideration. For example, a cost improvement of 5× could mean that the Raspberry Pi 3B+ configuration is 5× faster than the traditional server while having the same cost, or that the Raspberry Pi 3B+ configuration takes twice as long to run a query but costs 10× less. The break-even point of 1× (i.e., when both exhibit equal normalized performance) is denoted on each plot as a dotted black line, with values above the line indicating that the Raspberry Pi 3B+ configuration is better and numbers below the line indicating that the traditional server is better.

Again, the traditional servers actually have many other necessary hardware components (e.g., memory, persistent storage)

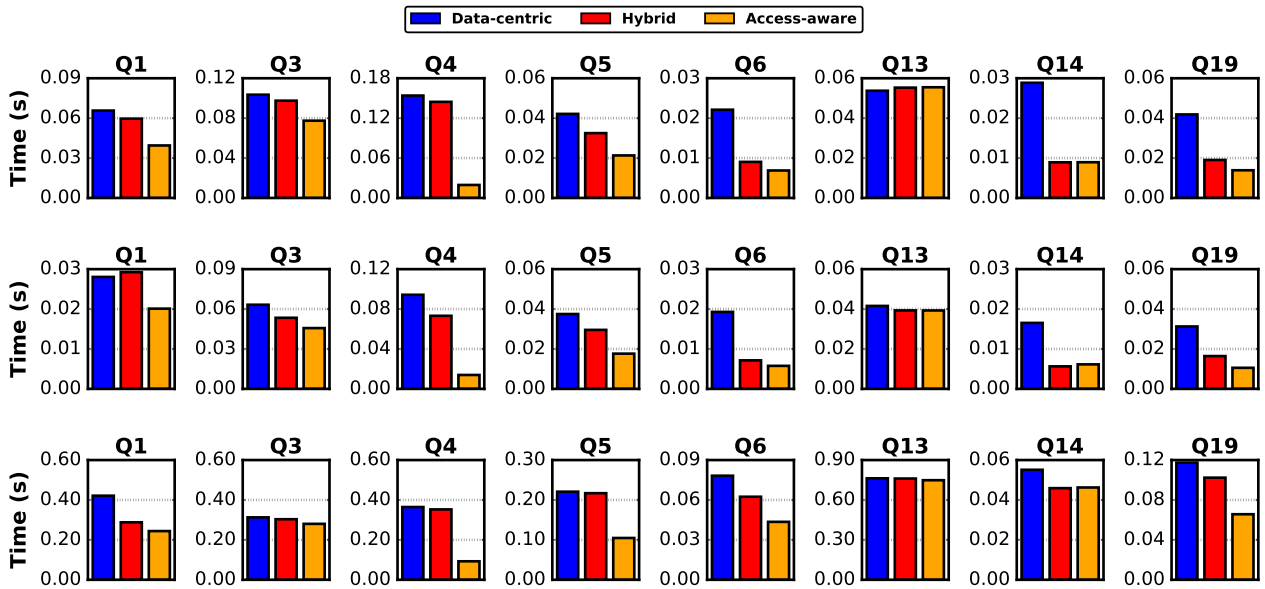


Fig. 4: Execution strategy results for `op-e5` (top), `op-gold` (middle), and Pi 3B+ (bottom)

that contribute greatly to both the overall cost and energy consumption. However, since the On-Premises servers were built for general-purpose use prior to this study, they do not include the most cost-effective or energy-efficient components. Our analyses therefore consider only the cost (i.e., MSRP) and energy consumption (i.e., TDP) of the CPU.

The result, then, is a pessimistic analysis; that is, a full cost comparison that includes all peripherals would actually be much better for WIMPI. While we could have leveraged this fact to make our results appear more favorable (e.g., artificially inflating total server costs by factoring in large and expensive memory), we believe that our current methodology yields a fair representation of the strengths and weaknesses of the Raspberry Pi 3B+.

A. Price Comparison

In the following, we analyze the results from our TPC-H benchmarks in terms of MSRP and hourly costs. We then conclude with a brief discussion of other cost metrics.

1) MSRP: Our first price analysis considers performance relative to the total cost of each comparison point, with runtimes normalized by the MSRP (Figure 5). Again, since the CPUs for the Cloud servers are custom SKUs for Amazon EC2, we cannot determine the MSRP and thus compare only against the On-Premises servers. Note that, since `op-e5` and `op-gold` are dual-socket, our calculation doubles the MSRP values from Table I.

For SF 1, the single Raspberry Pi 3B+ always outperforms the traditional servers, with a roughly 7–41× improvement over `op-e5` and 6–64× improvement over `op-gold`, with median values of 22× and 29×, respectively. The Raspberry Pi 3B+ has a 30% greater improvement over `op-gold` compared to `op-e5`, which suggests that the significantly more expensive Xeon Gold 6150 CPU does not offer commensurate performance benefits for this workload.

The comparison for SF 10 is similarly promising, with WIMPI demonstrating improvements on seven of the eight tested queries. For half of the queries (Q1, Q3, Q4, Q5), a smaller number of four (or in some cases even eight) nodes do not offer sufficient performance to exceed the break-even cost threshold, but adding enough nodes to produce the aforementioned performance jumps makes WIMPI between 2–8× more cost-effective. Adding more nodes beyond this point results in performance increases that keep pace with the additional costs, such that we observe no additional improvements in normalized performance at larger cluster sizes. In three of the other queries (Q6, Q14, Q19), we actually observe the opposite trend where additional nodes hurt the normalized performance.

Finally, in the case of Q13, the traditional servers are always better, irrespective of cluster size. Recall that this query executes using only a single node in our cluster, so additional nodes increase the total cost without reducing query runtime.

2) Hourly: Similar to our MSRP analysis, we normalize runtimes for the Cloud group using the hourly cost charged by Amazon EC2 [26]. For the Raspberry Pi 3B+, recall that we calculate the estimated hourly cost of $0.0004 using the maximum possible power draw and US national average price per kilowatt-hour [28]. The results appear in Figure 6.

In some ways, the particulars of the hourly cost comparison are less interesting than the MSRP analysis, since the Raspberry Pi 3B+ outperforms all Cloud servers for all queries in both the single-node (up to 10,000× improvement) and distributed (up to 1,200× improvement) settings. Even for Q13 in SF 10, where WIMPI was much worse for all cluster sizes

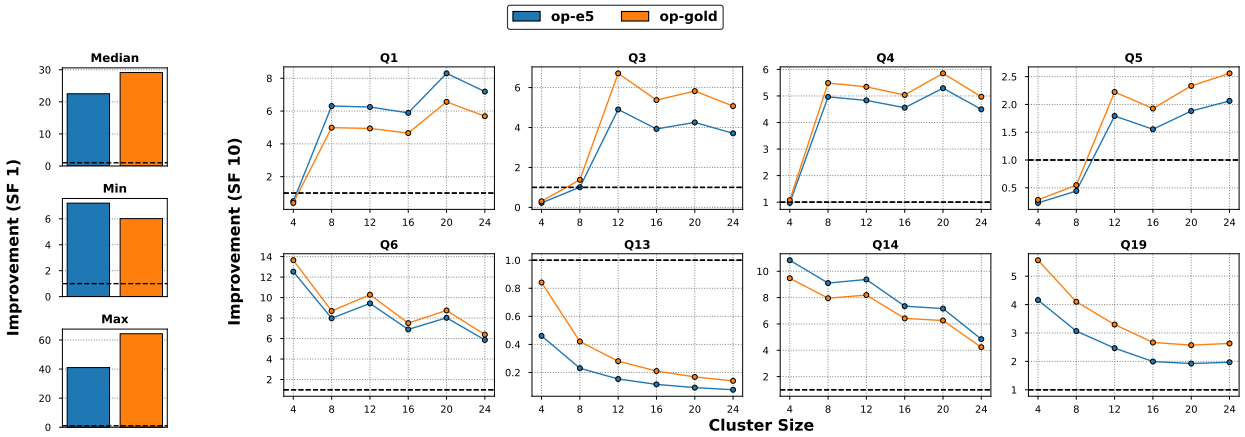


Fig. 5: MSRP comparison for SF 1 (left) and SF 10 (right)

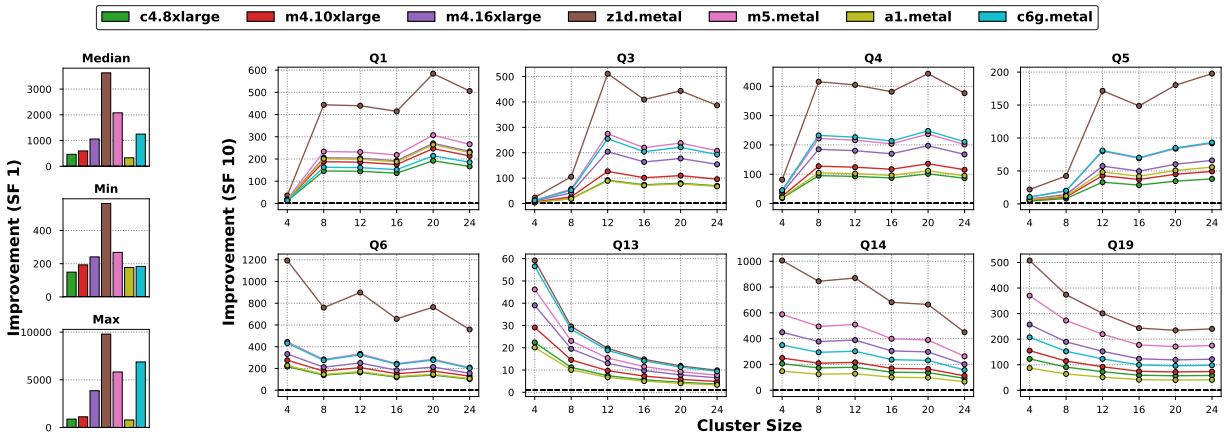


Fig. 6: Hourly cost comparison for SF 1 (left) and SF 10 (right)

in the MSRP analysis, we observed a 3–10× improvement over the Cloud servers in the worst case (24 nodes).

Despite the enormous cost savings, cloud deployments obviously have significant advantages in terms of ease of use, since cluster operation and maintenance is outsourced to the cloud provider. In the case of a cluster like WIMPI, on the other hand, one may need to manage dozens of SBC nodes. We discuss ease of use, as well as other important considerations, separately in Section III-C.

3) Other Metrics: So far, we have analyzed performance in terms of both MSRP and hourly cost, and the SBCs demonstrate clear advantages in these comparisons. However, many other common metrics exist for evaluating cost.

For example, the total cost of ownership (TCO) is a popular metric that has appeared in several similar evaluations [20], [23], [14], [21]. As stated earlier, this metric is difficult to assess accurately given the wide variability in terms of additional components, both for the traditional servers and the Raspberry Pi 3B+. Moreover, many of the peripherals required for the Raspberry Pi 3B+ (e.g., Ethernet cables, microSD cards) are relatively inexpensive compared to all of the hardware components necessary for a traditional server, which include memory, persistent storage, a motherboard, power supplies, and fans, among others.

An even larger challenge arises when attempting to incorporate electricity costs for power and cooling into a TCO analysis. In addition to large variability in price, the aforementioned components can also have huge differences in energy requirements. For example, servers today will typically have a large amount of memory [1], [2], but this type of provisioning might result in unnecessarily high energy consumption for workloads that do not require it.

For these reasons, we chose to forego a formal TCO analysis, which would have heavily favored the Raspberry Pi 3B+ due to much cheaper peripherals and significantly reduced energy costs.

B. Energy Comparison

The second key advantage of SBCs is their reduced power consumption. As explained, since the specifications for EC2 instances are not public, we evaluate only the energy consumption for the On-Premises servers.

1) Active: Figure 7 shows our TPC-H results normalized by energy consumption. Similar to past studies [40], [15], [41], we use the reported TDP (Table I) to estimate CPU energy consumption for the servers. Again, like our MSRP comparison, we do not consider the energy consumption for other server components, resulting in a pessimistic analysis for the Raspberry Pi 3B+.

The SF 1 results show that a single Raspberry Pi 3B+ offers between 2–22× better energy efficiency, with a median improvement of around 10×. The improvements for SF 10 are less pronounced but still promising: Raspberry Pi 3B+ has better energy efficiency on six of the eight queries, with the maximum improvements in the range of 5–6×.

These results contradict another recent study [14] that concluded SBCs should exhibit the best energy efficiency for memory-bound scan queries like Q1. However, given the extremely limited memory bandwidth of the Raspberry Pi 3B+, these types of queries end up executing for much longer relative to computationally intensive queries, leading to an underutilized CPU and greater sustained power draw compared to the traditional servers. Rather, as shown in our experiments, highly selective queries (e.g., Q6) that do not exhaust the memory bandwidth show the best speedup and, consequently, best improvement in energy consumption.

2) Idle: Since clusters often spend a significant amount of time idle [42], power consumption during these idle periods is an important consideration. Ideally, servers should be energy-proportional [42], such that the power draw is commensurate with the amount of work being performed. However, traditional servers almost always have very poor energy proportionality, since certain components like memory require constant power draw to operate, regardless of the current load. On the other hand, the Raspberry Pi 3B+ nodes that comprise WIMPI are highly energy-proportional, and they use very little power when idle.

Further, we believe that one of the key benefits of SBC clusters like WIMPI is the ability to add or remove resources at a very fine level of granularity in order to maximize performance while minimizing wasted energy. When not in use, individual Raspberry Pi 3B+ nodes could easily be turned off to save power. At the same time, should cluster utilization increase, SBCs can boot up much faster than traditional servers, allowing a cluster of SBCs to respond much more quickly to changes in demand.

3) Cooling: One final aspect of energy consumption that we do not directly measure is the cost of cooling, since isolating this cost for a single machine in a server room is impracticable. However, since cooling still represents one of the largest components of total operating costs [19], [29], we highlight some advantages of SBCs in this regard.

As previously mentioned, our WIMPI cluster does not require any external cooling infrastructure. The low energy consumption of the Raspberry Pi 3B+ allows WIMPI to be entirely air-cooled at normal room temperature, with spacers installed to ensure sufficient distance between stacked nodes for proper airflow.

If individual nodes were to become overheated, though, they would throttle the CPU, leading to degraded performance and potential node failures. While running our benchmarks, we monitored the CPU temperature for each Raspberry Pi 3B+ using the built-in monitoring tools. Even for experiments with sustained heavy load (e.g., the CPU microbenchmarks from Section II-C), the observed temperatures always remained within normal operating thresholds and never resulted in throttling. In settings where the Raspberry Pi 3B+ might be placed in a protective casing that obstructs airflow (e.g., for industrial use cases), overheating may become a problem, but we never encountered issues operating at normal room temperature with our cluster setup.

C. Other Factors

Overall, we have shown that currently available SBCs like the Raspberry Pi 3B+ can offer significant cost and energy savings for in-memory OLAP workloads. We also demonstrated that a cluster of Raspberry Pi 3B+ nodes can even maintain these same benefits while providing absolute query runtimes on par with traditional servers. However, operating a cluster like WIMPI comes with a variety of unique challenges, several of which we discuss in the following.

1) Memory Size: One main drawback of SBCs is their limited memory sizes. As mentioned, the Raspberry Pi 3B+ SBCs that we used in this study have only 1 GB of memory. While MonetDB's in-memory storage format allows TPC-H SF 10, including the base tables and all intermediate query results, to fit comfortably in the aggregate memory of just four WIMPI nodes, larger databases might exceed the capacity of a reasonably sized cluster.

Limited memory sizes are especially problematic for some popular distributed data processing frameworks (e.g., Hadoop [43], Spark [44]) that have high memory overheads. For example, when we tested Spark on WIMPI, we found that nearly half of the available 1 GB of memory was consumed by the JVM and Spark runtime, leaving only 500 MB for the base data and intermediate query results. Past studies [14], [21] that evaluated JVM-based systems encountered frequent crashes during their experiments, which may have informed their conclusion that SBCs are ill-suited for complex workloads.

However, just as memory prices continue to drop for traditional servers, SBCs are experiencing a similar trend. The Raspberry Pi 4B [27] already comes in a variant with 8 GB of memory, but these larger memory sizes are naturally more expensive. Additionally, more memory also means more energy consumption. Weighing these trade-offs is not straightforward, but they allow for the intriguing possibility of tailoring the node composition of SBC clusters to individual workloads.

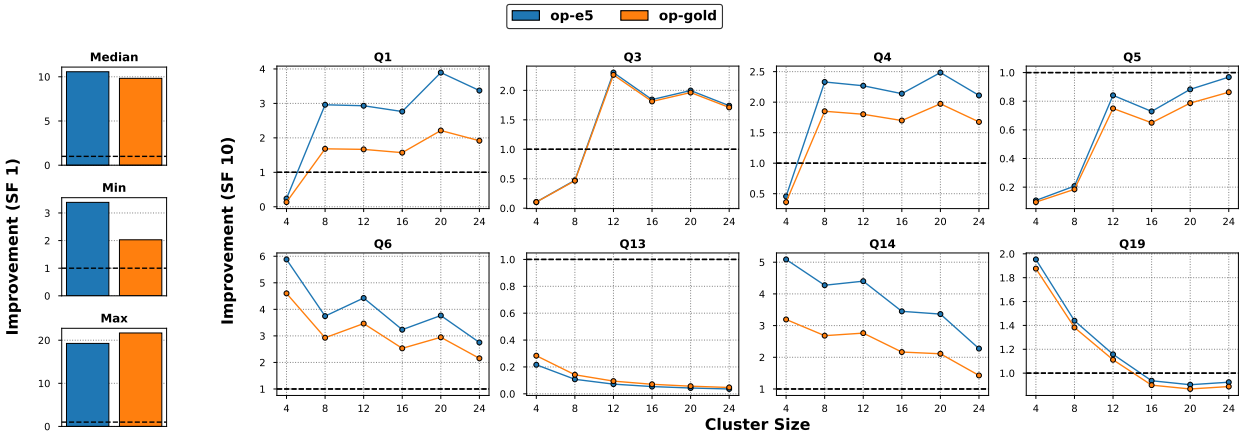


Fig. 7: Energy comparison for SF 1 (left) and SF 10 (right)

Of course, if the data becomes too large, the DBMS could always just spill to persistent storage, but the microSD cards used in the WIMPI cluster have extremely limited bandwidth. Unfortunately, the solution is not as simple as upgrading the hardware. Equipping all nodes with SSDs, for instance, would drastically change the price and energy considerations for the cluster, and the SSD would also face the same bandwidth limitations as network I/O due to the shared bus.

Therefore, one interesting alternative is a network-attached memory (NAM) architecture [45], where a single traditional server would host a large memory pool that can be remotely accessed by the Raspberry Pi 3B+ nodes. This configuration also provides unique query processing opportunities in which the server could perform tasks that require a large amount of memory, such as an aggregation with many distinct keys or performing a join. We believe that these types of hybrid clusters [46], [47], [21] that combine the benefits of both platforms represent a strong avenue for future investigation.

2) Memory Bandwidth: A related issue is the limited aggregate memory bandwidth of the WIMPI cluster compared to the traditional servers. As mentioned, WIMPI has a total bandwidth of about 48 GB/s, and our experimental results (Section II-D) demonstrated the impact of this shortcoming on query performance.

One possibility is to leverage more aggressive compression to help alleviate the memory bandwidth bottleneck. While in-memory DBMSs have traditionally favored computationally lightweight techniques at the expense of greater memory consumption (e.g., fixed-width dictionary encoding), the comparatively excellent CPU performance could open the door for algorithms previously considered too costly.

Similarly, DBMSs will often avoid performing redundant work by carefully materializing intermediate results. However, to reduce memory bandwidth pressure, completely recomputing some of these intermediates might actually be better than caching them.

3) Usability: For any real-world deployment, ease of use remains one of the largest factors in the decision-making process. While running a single machine is always simpler than managing an entire cluster, we found that WIMPI was surprisingly easy to set up and use.

The first task was to distribute the data across nodes in the cluster. For WIMPI, we chose to use HDFS, though many other distributed file systems exist. In recent years, a huge number of distributed data processing frameworks have also emerged, making it easier than ever to run a wide variety of data analytics workloads on a cluster.

As previously explained, we selected MonetDB for our TPC-H experiments. Since no pre-built release binary existed for several of our target hardware platforms, we had to build from source, which was an extremely straightforward process. Although MonetDB comes with built-in distributed query processing capabilities, we found that the distributed query planner was not particularly advanced and implemented only a few basic optimizations. For example, the planner would push down only selections and aggregations to remote nodes while sending large intermediate results to a single node in order to perform all joins locally. Although this execution strategy worked fine for queries like Q1 and Q6 that can simply perform parallel scans of the `lineitem` table, the entire WIMPI cluster ground to a halt when attempting to transmit large intermediate results to a single Raspberry Pi 3B+ node due to the limited network bandwidth and single-node memory capacity.

We therefore wrote a very simple driver program using the MonetDB Python client API that runs on one of the WIMPI nodes and aggregates partial results after join processing is performed locally on each node. Due to the substantial reduction in network traffic, this straightforward optimization decreased runtimes for many queries by several orders of magnitude, and we expect future development of MonetDB's distributed processing functionality will incorporate more advanced techniques (e.g., distributed joins) that can achieve even better

performance. Other related features, such as transparent table partitioning and replication, would also go a long way to improve usability.

4) Reliability: Finally, we address the common misconception that SBCs are much less reliable than server-grade hardware. In the case of the Raspberry Pi 3B+ SBCs used in this study, ongoing design improvements have made current models much more stable than previous generations, despite their enduring $35 price tag.

In our experience, node failures in the distributed setting almost always resulted from virtual memory thrashing; WIMPI nodes would become generally unresponsive when either the database size or intermediate query results exceeded the available memory. Again, this problem stems from the extremely limited bandwidth of the microSD cards used in the WIMPI cluster for persistent storage. We were able to resolve the issue by disabling the swap space on all nodes to prevent inadvertent thrashing, which allowed isolated out of memory errors to occur without crashing an entire node.

After turning off swapping, we did not encounter any further node failures during the entire duration of our benchmarking study, nor did we run into any hardware failures. Our experience comports with another similar study [21], which reported only a single failure of a breakout board during ten months of operating a wimpy cluster comprised of 35 Intel Edison SBCs, with no failures of the actual SBCs themselves. Although anecdotal, these experiences support the idea that modern SBCs are actually quite robust, suggesting that fears about reliability might be somewhat outdated.

IV. Related Work

In this section, we provide a consolidated summary of the related work that was interspersed throughout the paper. Broadly, our study has overlap in the areas of (1) wimpy clusters and (2) energy efficiency.

A. Wimpy Clusters

The idea of replacing server-grade CPUs with a cluster of wimpy nodes is not new, with numerous examples of research, commercial, and even hobbyist prototypes. However, past work typically dismisses wimpy clusters as unsuitable for complex workloads, instead focusing on other use cases.

For instance, FAWN [20] investigated the use of wimpy nodes in the context of a key-value store, which involves many random accesses to small data items (i.e., point lookups) rather than the large scans typical of OLAP workloads. Others have explored the feasibility of leveraging wimpy nodes for web servers [21], search engines [22], and distributed data processing frameworks (e.g., Hadoop [43]) for I/O-bound batch processing [48], [14], [21]. In fact, some have even argued that wimpy nodes are completely unable to handle computationally intensive workloads [23], [21].

On the contrary, we have shown that recent hardware advancements have yielded vast performance improvements for currently available SBCs like the Raspberry Pi 3B+. Our microbenchmarks demonstrate that these so-called "wimpy" nodes actually have comparable core-to-core performance with some server-grade CPUs for computationally intensive tasks, and the TPC-H experiments show that the WIMPI cluster can even offer competitive performance for in-memory OLAP workloads while maintaining benefits in terms of cost and energy consumption.

Finally, as mentioned, some existing work has explored wimpy nodes in the context of heterogeneous clusters with virtualization over a collection of different machine types [46], [47] or architectures where a traditional server manages many wimpy worker nodes [21]. All of these approaches attempt to create a more balanced computing infrastructure, which is a promising direction that we intend to explore in the future.

B. Energy Efficiency

Energy consumption has become an increasingly important topic, and several studies have investigated the energy usage of DBMSs [40], [15], [17], [18]. Some have advocated for making DBMSs energy-aware by treating energy consumption as a first-class citizen in the optimizer, with the end goal of trading off energy for performance [19], [16], [49], [41], [50]. In this study, we considered only straightforward distributed query processing techniques, but the incorporation of energy consumption estimates during the query planning stage is an interesting idea for future work.

Other approaches have looked into achieving energy proportionality [42], [29], [51] by switching individual components (e.g., CPU [52], memory [53], disk [54]) or entire nodes in a cluster [55], [56], [57], [35], [36] into low-power states, or even by completely powering them off. We believe that these types of optimizations represent a major opportunity for SBC clusters like WIMPI, since they enable much more fine-grained resource control than in a traditional server.

V. Conclusion

This paper made the case that inexpensive SBCs like the Raspberry Pi 3B+ can serve as a viable alternative to high-end hardware for in-memory OLAP workloads. Our microbenchmarks demonstrated that a single Raspberry Pi 3B+ is surprisingly competitive with server-grade CPUs in a core-to-core performance comparison, and our analysis of the reported TPC-H results showed that a cluster of Raspberry Pi 3B+ nodes like WIMPI can achieve reasonable runtimes at a fraction of the cost, as well as other long-term advantages in terms of reduced energy consumption and cooling costs.

We believe that these results provide a solid foundation for continued investigation, as we have identified many of the core strengths and weaknesses of SBCs relative to server-grade CPUs. In the near future, we plan to extend our study with other computationally intensive workloads, in particular machine learning. Consequently, we also plan to evaluate how the Raspberry Pi 3B+ compares to hardware traditionally used for machine learning tasks (e.g., GPUs) and specialized accelerators (e.g., TPUs).

ACKNOWLEDGMENTS

We would like to thank the anonymous reviewers for their helpful feedback. This work received support from the AWS Cloud Credits for Research Program.

REFERENCES

[1] A. Crotty, A. Galakatos, K. Dursun, T. Kraska, U. Çetintemel, and S. B. Zdonik, "Tupleware: "Big" Data, Big Analytics, Small Clusters," in *CIDR*, 2015.
[2] A. Crotty, A. Galakatos, K. Dursun, T. Kraska, C. Binnig, U. Çetintemel, and S. Zdonik, "An Architecture for Compiling UDF-centric Workflows," *PVLDB*, vol. 8, no. 12, pp. 1466–1477, 2015.
[3] "Amazon EC2 Instance Types," https://aws.amazon.com/ec2/instance-types/.
[4] "E7-8880 v3," https://ark.intel.com/content/www/us/en/ark/products/84683/intel-xeon-processor-e7-8880-v3-45m-cache-2-30-ghz.html.
[5] "Google Cloud Machine Types," https://cloud.google.com/compute/docs/machine-types.
[6] "Microsoft Azure Virtual Machines," https://azure.microsoft.com/en-us/pricing/details/virtual-machines/series/.
[7] "Raspberry Pi 3 Model B+," https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/.
[8] D. O'Keeffe, T. Salonidis, and P. R. Pietzuch, "Frontier: Resilient Edge Processing for the Internet of Things," *PVLDB*, vol. 11, no. 10, pp. 1178–1191, 2018.
[9] S. Zeuch, A. Chaudhary, B. D. Monte, H. Gavriilidis, D. Giouroukis, P. M. Grulich, S. Breß, J. Traub, and V. Markl, "The NebulaStream Platform for Data and Application Management in the Internet of Things," in *CIDR*, 2020.
[10] J. Hülsmann, J. Traub, and V. Markl, "Demand-based Sensor Data Gathering with Multi-Query Optimization," *PVLDB*, vol. 13, no. 12, pp. 2801–2804, 2020.
[11] W. Xu, O. Curé, and P. Calvez, "SuccinctEdge: A Succinct RDF Store for Edge Computing," *PVLDB*, vol. 13, no. 12, pp. 2857–2860, 2020.
[12] C. Wang, X. Huang, J. Qiao, J. Wang, J. Sun, K. Mcgrail, J. Feinauer, J. Zhang, P. Wang, J. Zhang, R. Kang, T. Jiang, L. Rui, and J. Yuan, "Apache IoTDB: Time-series Database for Internet of Things," *PVLDB*, vol. 13, no. 12, pp. 2901–2904, 2020.
[13] T. Mühlbauer, W. Rödiger, R. Seilbeck, A. Reiser, A. Kemper, and T. Neumann, "One DBMS for all: the Brawny Few and the Wimpy Crowd," in *SIGMOD*, 2014, pp. 697–700.
[14] D. Loghin, B. M. Tudor, H. Zhang, B. C. Ooi, and Y. M. Teo, "A Performance Study of Big Data on Small Nodes," *PVLDB*, vol. 8, no. 7, pp. 762–773, 2015.
[15] M. Poess and R. O. Nambiar, "Energy Cost, The Key Challenge of Today's Data Centers: A Power Consumption Analysis of TPC-C Results," *PVLDB*, vol. 1, no. 2, pp. 1229–1240, 2008.
[16] W. Lang and J. M. Patel, "Towards Eco-friendly Database Management Systems," in *CIDR*, 2009.
[17] J. Meza, M. A. Shah, P. Ranganathan, M. Fitzner, and J. Veazey, "Tracking the Power in an Enterprise Decision Support System," in *ISLPED*, 2009, pp. 261–266.
[18] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah, "Analyzing the Energy Efficiency of a Database Server," in *SIGMOD*, 2010, pp. 231–242.
[19] G. Graefe, "Database Servers Tailored to Improve Energy Efficiency," in *SETMDM@EDBT*, 2008, pp. 24–28.
[20] D. G. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan, "FAWN: A Fast Array of Wimpy Nodes," in *SOSP*, 2009, pp. 1–14.
[21] Y. Zhao, S. Li, S. Hu, H. Wang, S. Yao, H. Shao, and T. F. Abdelzaher, "An Experimental Evaluation of Datacenter Workloads On Low-Power Embedded Micro Servers," *PVLDB*, vol. 9, no. 9, pp. 696–707, 2016.
[22] V. J. Reddi, B. C. Lee, T. M. Chilimbi, and K. Vaid, "Web Search Using Mobile Cores: Quantifying and Mitigating the Price of Efficiency," in *ISCA*, 2010, pp. 314–325.
[23] W. Lang, J. M. Patel, and S. Shankar, "Wimpy Node Clusters: What About Non-Wimpy Workloads?" in *DaMoN*, 2010, pp. 47–55.
[24] "E5-2660 v2," https://ark.intel.com/content/www/us/en/ark/products/75272/intel-xeon-processor-e5-2660-v2-25m-cache-2-20-ghz.html.
[25] "Gold 6150," https://ark.intel.com/content/www/us/en/ark/products/120490/intel-xeon-gold-6150-processor-24-75m-cache-2-70-ghz.html.
[26] "Amazon EC2 Pricing," https://aws.amazon.com/ec2/pricing/on-demand/.
[27] "Raspberry Pi 4 Model B," https://www.raspberrypi.org/products/raspberry-pi-4-model-b/.
[28] "Electric Power Monthly Estimates," https://www.eia.gov/electricity/monthly/epm_table_grapher.php?t=epmt_5_6_a.
[29] J. R. Hamilton, "Internet-Scale Data Center Power Efficiency," in *CIDR*, 2009.
[30] H. J. Curnow and B. A. Wichmann, "A Synthetic Benchmark," *Comput. J.*, vol. 19, no. 1, pp. 43–49, 1976.
[31] R. Weicker, "DHRYSTONE: A Synthetic Systems Programming Benchmark," *Commun. ACM*, vol. 27, no. 10, pp. 1013–1030, 1984.
[32] "sysbench," https://github.com/akopytov/sysbench.
[33] "iperf," http://manpages.ubuntu.com/manpages/bionic/man1/iperf.1.html.
[34] "MonetDB," https://www.monetdb.org/.
[35] W. Lang, S. Harizopoulos, J. M. Patel, M. A. Shah, and D. Tsirogiannis, "Towards Energy-Efficient Database Cluster Design," *PVLDB*, vol. 5, no. 11, pp. 1684–1695, 2012.
[36] D. Schall and T. Härder, "Energy-proportional Query Execution using a Cluster of Wimpy Nodes," in *DaMoN*, 2013, pp. 1–6.
[37] P. Menon, A. Pavlo, and T. C. Mowry, "Relaxed Operator Fusion for In-Memory Databases: Making Compilation, Vectorization, and Prefetching Work Together At Last," *PVLDB*, vol. 11, no. 1, pp. 1–13, 2017.
[38] A. Crotty, A. Galakatos, and T. Kraska, "Getting Swole: Generating Access-Aware Code with Predicate Pullups," in *ICDE*, 2020, pp. 1273–1284.
[39] P. A. Boncz, T. Neumann, and O. Erling, "TPC-H Analyzed: Hidden Messages and Lessons Learned from an Influential Benchmark," in *TPCTC*, 2013, pp. 61–76.
[40] S. Rivoire, M. A. Shah, P. Ranganathan, and C. Kozyrakis, "JouleSort: A Balanced Energy-Efficiency Benchmark," in *SIGMOD*, 2007, pp. 365–376.
[41] Z. Xu, Y. Tu, and X. Wang, "Exploring Power-Performance Tradeoffs in Database Systems," in *ICDE*, 2010, pp. 485–496.
[42] L. A. Barroso and U. Hölzle, "The Case for Energy-Proportional Computing," *Computer*, vol. 40, no. 12, pp. 33–37, 2007.
[43] "Apache Hadoop," https://hadoop.apache.org.
[44] "Apache Spark," https://spark.apache.org/.
[45] C. Binnig, A. Crotty, A. Galakatos, T. Kraska, and E. Zamanian, "The End of Slow Networks: It's Time for a Redesign," *PVLDB*, vol. 9, no. 7, pp. 528–539, 2016.
[46] N. Tolia, Z. Wang, M. Marwah, C. Bash, P. Ranganathan, and X. Zhu, "Delivering Energy Proportionality with Non Energy-Proportional Systems - Optimizing the Ensemble," in *HotPower*, 2008.
[47] B. Chun, G. Iannaccone, G. Iannaccone, R. H. Katz, G. Lee, and L. Niccolini, "An Energy Case for Hybrid Datacenters," *SIGOPS OSR*, vol. 44, no. 1, pp. 76–80, 2010.
[48] A. M. Caulfield, L. M. Grupp, and S. Swanson, "Gordon: Using Flash Memory to Build Fast, Power-efficient Clusters for Data-intensive Applications," in *ASPLOS*, 2009, pp. 217–228.
[49] S. Harizopoulos, M. A. Shah, J. Meza, and P. Ranganathan, "Energy Efficiency: The New Holy Grail of Data Management Systems Research," in *CIDR*, 2009.
[50] W. Lang, R. Kandhan, and J. M. Patel, "Rethinking Query Processing for Energy Efficiency: Slowing Down to Win the Race," *IEEE Data Eng. Bull.*, vol. 34, no. 1, pp. 12–23, 2011.
[51] A. S. Szalay, G. C. Bell, H. H. Huang, A. Terzis, and A. White, "Low-Power Amdahl-Balanced Blades for Data Intensive Computing," *SIGOPS OSR*, vol. 44, no. 1, pp. 71–75, 2010.
[52] D. Meisner, B. T. Gold, and T. F. Wenisch, "PowerNap: Eliminating Server Idle Power," in *ASPLOS*, 2009, pp. 205–216.
[53] X. Fan, C. S. Ellis, and A. R. Lebeck, "Memory Controller Policies for DRAM Power Management," in *ISLPED*, 2001, pp. 129–134.
[54] Q. Zhu, Z. Chen, L. Tan, Y. Zhou, K. Keeton, and J. Wilkes, "Hibernator: Helping Disk Arrays Sleep through the Winter," in *SOSP*, 2005, pp. 177–190.
[55] J. S. Chase, D. C. Anderson, P. N. Thakar, A. Vahdat, and R. P. Doyle, "Managing Energy and Server Resources in Hosting Centers," in *SOSP*, 2001, pp. 103–116.
[56] W. Lang and J. M. Patel, "Energy Management for MapReduce Clusters," *PVLDB*, vol. 3, no. 1, pp. 129–139, 2010.
[57] D. Schall and V. Hudlet, "WattDB: An Energy-Proportional Cluster of Wimpy Nodes," in *SIGMOD*, 2011, pp. 1229–1232.