Methods for Quantifying Energy Consumption in TPC-H

Meikel Poess
Oracle
Redwood Shores, California
meikel.poess@oracle.com

Da Qi Ren
Futurewei Technologies
Santa Clara, California
Daqi.Ren@huawei.com

Tilmann Rabl
TU Berlin
Berlin, Germany
rabl@tu-berlin.de

Hans-Arno Jacobsen
TU Munich
Munich, Germany
jacobsen@in.tum.de

ABSTRACT

Historically, performance and price-performance of computer systems have been the key purchasing arguments for customers. However, with rising energy costs and increasing power consumption due to the ever-growing demand for compute power (servers, storage, networks), electricity bills have become a significant expense for today's data centers. In order to measure energy consumption in standardized ways, the Standard Performance Evaluation Corporation (SPEC) has developed a benchmark dedicated to measuring the power consumption of single servers (SPECpower_ssj2008), while the Transaction Processing Performance Council (TPC) and the Storage Performance Council (SPC) have developed general specifications that govern how energy is measured for any of its benchmarks. Energy reporting is optional in TPC and SPC results. While there are close to 600 SPECpower_ssj2008 results, there have been only three TPC and no SPC benchmark results published that report energy consumption. In this paper, we argue that the low number of TPC publications is due to the large setups required in TPC benchmarks and the, subsequently, complicated measurement setup. Running on a typical big data setup we evaluate two alternative methods to quantify energy consumption during TPC-H's multi-user runs, namely by taking measurements of on-chip power sensors controlled through Intelligent Platform Management Interface and by estimating power consumption via the nameplate power consumption method. We compare these later two methods with power measurements taken from external power meters as required by SPEC and TPC benchmarks.

CCS CONCEPTS

• **Information systems → Database performance evaluation**;

KEYWORDS

Data Warehouse, Benchmarking, Energy Consumption Estimation

ICPE '18, April 9–13, 2018, Berlin, Germany

ACM ISBN 978-1-4503-5095-2/18/04…$15.00
https://doi.org/10.1145/3184407.3184429

ACM Reference Format:
Meikel Poess, Da Qi Ren, Tilmann Rabl, and Hans-Arno Jacobsen. 2018. Methods for Quantifying Energy Consumption in TPC-H. In *ICPE '18: ACM/SPEC International Conference on Performance Engineering, April 9–13, 2018, Berlin, Germany*. ACM, New York, NY, USA, Article 4, 12 pages. https://doi.org/10.1145/3184407.3184429

1 INTRODUCTION

In light of the increasing power consumption of data centers industry standard organizations such as the Transaction Processing Performance Council (TPC) [12], the Standard Performance Evaluation Corporation (SPEC) [10] and the Storage Performance Council (SPC) [9] have developed methodologies to measure energy consumption of computer systems. All of these consortia aim at standardizing power consumption measurement for performance benchmarks to aid IT departments in their purchase decision process. Their approaches, however, differ. Some organizations developed specialized benchmarks (SPEC [11]) while others added energy metrics to existing benchmarks (TPC,SPC [8]).

SPEC has been the front-runner by announcing the first industry standard benchmark, SPECpower_ssj2008, to measure power consumption in relation to performance for server-class computers in 2007. SPECpower_ssj2008 measures the performance of a Java based middle tier emulating client and database tiers. By emulating many components, that would otherwise be painful to setup, SPECpower_ssj2008 can be run on a single server without large hardware installations. It measures processor and memory performance, ignoring disk and network I/O. On the software side it measures the performance of the Java Virtual Machine, just-in-time compilation, garbage collection, user threads and some aspects of the operating system.

SPC, on the other hand, published two optional energy extensions to its existing benchmarks, SPC-1C and SPC-1. They were released in June 2009 and October 2009, respectively. The underlying performance tests for SPC-1C/E and SPC-1/E are identical to those used in their parent benchmarks. The energy extensions focus on defining appropriate enhancements to these core benchmark components, such as approving power meters to measure power consumption, defining disclosure requirements and define appropriate power metrics.

TPC took a similar approach to measuring energy consumption. Since it already has a large suite of benchmarks modeled after real-life scenarios such as TPC-C and TPC-E for online transaction processing, TPC-H for data warehousing, TPC-DS, TPCx-BB, and

TPCx-HS for big data, TPCx-V and TPC-VMS for virtualization, TPC-DI for data integration and TPCx-IoT for the Internet of Things, the TPC developed the *energy specification*. The energy specification is, similar to TPC's pricing specification, a *common* specification. Common specification are intended to supplement existing TPC benchmarks by specifying the amendments necessary to measure and report energy metrics in addition to the other metrics within each of the individual benchmarks. Before the energy specification can be used in conjunction with a benchmark, that benchmark need to define energy specific terms in its specification. These terms are necessary as each TPC benchmark defines its own metric and time measurements. So far only the TPC-E specification has been amended with these terms. The TPC provides a software package, called the TPC-Energy Measurement System (EMS) to aid in implementing the TPC Energy Specification. It includes modules to interface with power instrumentation tools, to log and report power and temperature.

While there are close to 600 SPECpower_ssj2008 results, there have been only three TPC benchmark results published with the optional energy consumption metric and no SPC benchmark results with the optional energy consumption metric. In this paper we argue that the low number of TPC publications is due to the large setups required in TPC benchmarks and the, subsequently, complicated measurement setup.

Using a typical big data setup, an 6 node cluster running Cloudera's Impala engine, we evaluate two alternative methods to quantify energy consumption in large setups, namely by taking measurements of on-chip power sensors controlled through Intelligent Platform Management Interface (IPMI) and by estimating power consumption via the *nameplate* power consumption method[2, 6, 7]. The nameplate value of computing equipment is the rated maximum power consumption of the equipment. It is a conservative power consumption estimate. We compare these later two methods with power measurements taken from external power meters as required by SPEC and TPC benchmarks.

The main contributions of this paper are best summarized as follows:

(1) We amend TPC-H [5] with energy specific terms so that the energy specification can be used with TPC-H;
(2) We apply the nameplate power consumption model to our hardware setup;
(3) We quantify energy consumption of a six node cluster running a 300G TPC-H database with three methodologies: external power meters, IPMI and power estimation using the nameplate power model;
(4) And we analyze the the power results of the above methods.

The remainder of our paper is organized as follows. Section 2 discusses the important characteristics of TPC-H, proposes changes to the TPC-H specification that allows for energy measurements and develops the comparison metrics that will be used to analyze quantitative energy methods compared in this study. We also describe necessary modifications to queries so that they run in Impala. Section 3 describes the hardware and software setup of the system that we are using for our experiments. Section 4 describes the three quantitative methods we use in our paper to determine energy consumption during TPC-H runs: (i) power meters, (ii) IPMI and (iii) analytical power consumption model using nameplate power consumption. Section 5 we present our findings of measuring power consumption using our three quantitative methods and we conclude in Section 6.

2 WORKLOAD

This section provides an overview of TPC-H including a brief workload characterization that is useful to understand the following sections.

2.1 Workload Characteristics

Generally, analytical workloads can be divided into parallel operations of four distinct types: initial load, queries, incremental load and auxiliary data creation/maintenance operations. These types can be executed in single and multi-user modes. The single-user mode stresses a system's ability to parallelize operations across all available system resources to answer a given request in the least amount of time. The multi-user mode stresses the system's ability to schedule requests from multiple concurrent users to optimally utilize all system resources with the overall aim of increasing system throughput.

TPC-H [5], developed by the Transaction Processing Performance Council, covers both single- and multi-user runs. There are other benchmarks like TPC-DS that also cover both single- and multi-user runs. However, the software stack we are using does not support all queries included in TPC-DS and using the well studied TPC-H benchmark gives the reader an instant understanding of our experiments. TPC-H is based on a relatively simple, yet powerful 3rd NF schema. It allows query execution of various execution paths. The access paths in a 3rd NF analytical workload are often dominated by large hash or sort-merge joins, but conventional index driven joins are also common. Large aggregations, which often include large sort operations, are widespread in TPC-H. This diversity imposes challenges both on hardware and software systems. High sequential I/O-throughput (large I/O operations) is critical to excel in large hash-join operations.

TPC-H's execution rules and main performance metric give equal importance to the single- and multi-user runs, while ignoring load time. The main performance metric (QphH) is calculated as the geometric mean of elapsed times collected during the single- and multi-user tests with concurrent users (see Figure 2) and adjusted using the raw database size specified with the Scale Factor SF. Scale factors allowed for publication are $SF \in \{1, 10, 30, 100, 300, 1000, 3000, 10000, 100000\}$. TPC-H mandates a minimum number of concurrent streams (S), while it defines no upper bound. The number of concurrent query and update streams is identical. Each user in the multi-user executes the 22 queries in a different permutation. The permutations are generated by the query generator *qgen*.

$$S(SF) = \begin{cases} log_{10}(SF) * 2 + 1 & \text{if } SF \in \{10, 100, 1000, 10000, 100000\} \\ (log_{10}(\frac{SF}{3}) + 1) * 2 & \text{otherwise} \end{cases} \quad (1)$$

Additionally, TPC-H's execution rules allow for the deferral of the update stream until all query streams are finished.

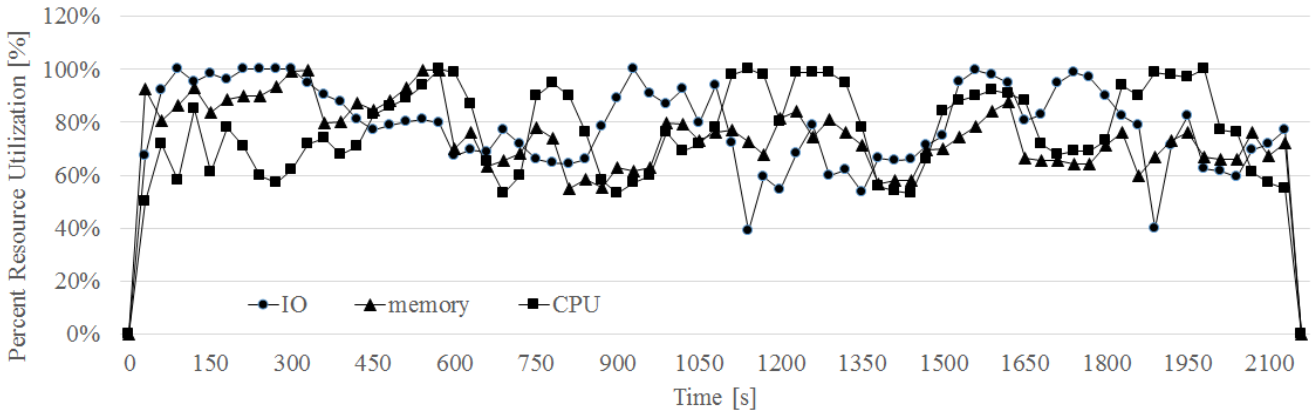

Figure 1: Oscillating resource utilization during a TPC-H throughput run

Each decision support query has its own hardware resource utilization pattern, which is unique to the way it is executed on a particular system. On a symmetrical multi processor (SMP) system the resources that are considered most important, especially when sizing a system for a particular workload, are processor (CPU), reads/writes (I/O) from/to the disk storage subsystem, disks, controllers, and memory. Big data workloads tend to be mostly read-only; however, join operations of large tables, aggregations with a large number of groups and sort operations of large data sets that exceed the internally available amount of memory result in write and read operations to temporary storage.

Except for Query 1 (CPU bound) and Query 6 (I/O bound) TPC-H queries exhibit an oscillating system resource pattern. Many queries join multiple tables and aggregate over a large data set. For an in-depth analysis of TPC-H queries see [1]. Due to their complex nature TPC-H queries do not exhaust all resources of a system during the entire run time of the query. For instance, a hash join is typically CPU bound during the build phase of its hash table and I/O bound during its probe phase. Consequently, a system consumes more power in the storage subsystem during some time of the single-user test and more CPU power during other times of the single-user test. To be able to apply the nameplate power consumption model two key requirements have to be met. Firstly, only workloads that observe steady state can be used. The second requirement is system balance. Depending on the application and system, an optimal component ratio has to be maintained to keep all components (CPU, disks, controllers etc.) utilized during the measurement interval. If a system does not have the optimal ratio between these components, the power consumption model will not produce accurate estimates for the same reason that the system needs to be fully utilized.

The multi-user test alleviates this effect, because the simultaneous execution of different queries cause the overlapping of different query execution phases. Because of many queries being executed at the same time, each query is likely to be in a different execution phase causing the resources of a system to be utilized in a more balanced way. That is, low utilization of resource R_1 in a query execution of one user might overlap with high utilization of R_1 in another query of another user. Figure 1 shows a multi-user run with 8 users against a SF=1000 TPC-H database. While there are still phases where the system resources drop to 60% the average CPU utilization during this run is 75%, the average IO utilization is 77% and the average memory utilization is 73%.

Traditionally the single-user mode has been mostly used in dedicated batch windows, for example, for data cleansing and the creation of auxiliary data structures. This batch window is becoming less important in today's big data deployments. Big data systems, many of which are based on the Hadoop ecosystem, trade data ownership with flexibility and availability. Traditionally, only one system had control over a given data set, namely the DBMS. This control enabled the DBMS to use techniques and algorithms to implement performance enhancements and to enforce data correctness that rely on persistent auxiliary data structures. Big data systems, on the other hand, follow an open data approach, in which all products in its ecosystem are able to access and modify the same full-fidelity data sets. This approach eliminates the costly process of copying and converting data into different formats and allows for immediate data availability, while rendering traditional DBMS concepts, such as auxiliary data structures, impractical because the big data system is not able to invalidate and, therefore, not able to guarantee correctness.

2.2 Modifications to the TPC-H queries for Impala

In order to run the TPC-H queries on the version of Impala that ships with Cloudera 5, we had to make several modifications to almost all queries. A summary of our changes is listed in Table 1. Most changes were related to the handling of date arithmetic, string and other arithmetic functions. However, we needed to rewrite Query 15, Query 18 and Query 21.

2.2.1 Query 15. The changes to Query 15 are summarized in Listing 2. Impala was not able to execute the original query (see Listing 1) because of the sub-query that returns the maximum total_revenue. Because of this being an uncorrelated sub-query, it only needs to be executed once. Hence, rewriting it as a create table as select statement (CTAS) is a reasonable approach. It does not

Table 1: Modifications to TPC-H queries to run on Impala

Qry	Original	Modifications
1	l_shipdate<=date '1998-12-01' - interval ':1' day	l_shipdate<=date_add(cast('1998-12-01' as timestamp),interval -:1 days)
2	r_name = ':3'	r_name like ':3%'
3	c_mktsegment = ':1'	c_mktsegment like ':1%'
	o_orderdate < date ':2'	o_orderdate < cast(':2' as timestamp)
	l_shipdate > date ':2'	l_shipdate > cast(':2' as timestamp)
4	o_orderdate >= date ':1'	o_orderdate >= cast(':1' as timestamp)
	o_orderdate < date ':1' + interval '3' month	o_orderdate < date_add(cast(':1' as timestamp),interval 3 months)
5	r_name = ':1'	r_name like ':1%'
	o_orderdate >= date ':2'	o_orderdate >= cast(':2' as timestamp)
	o_orderdate < date ':2' + interval '1' year	and o_orderdate < date_add(cast(':2' as timestamp), interval 12 months)
6	l_shipdate >= date ':1'	l_shipdate >= cast(':1' as timestamp)
	l_shipdate < date ':1' + interval '1' year	l_shipdate < date_add(cast(':1' as timestamp),interval 24 months)
7	(n1.n_name = ':1' and n2.n_name = ':2')	(n1.n_name like ':1%' and n2.n_name like ':2%')
	(n1.n_name = ':2' and n2.n_name = ':1')	(n1.n_name like ':2%' and n2.n_name like ':1%')
	l_shipdate between date '1995-01-01' and	l_shipdate between cast('1995-01-01' as timestamp) and
	date '1996-12-31'	cast('1996-12-31' as timestamp)
8	r_name = ':2'	and r_name like ':2%'
	o_orderdate between date '1995-01-01' and	o_orderdate between cast('1995-01-01' as timestamp) and
	date '1996-12-31'	cast('1996-12-31' as timestamp)
	p_type = ':3'	p_type like ':3%'
10	o_orderdate >= date ':1'	o_orderdate >= cast(':1' as timestamp)
	o_orderdate < date ':1' + interval '3' month	o_orderdate < date_add(cast(':1' as timestamp),interval 3 months)
	l_returnflag = 'R'	l_returnflag like 'R%'
11	n_name = ':1'	n_name like ':1%'
12	when o_orderpriority = '1-URGENT'	when o_orderpriority like '1-URGENT%'
	or o_orderpriority = '2-HIGH'	or o_orderpriority like '2-HIGH%'
	when o_orderpriority <> '1-URGENT'	when o_orderpriority not like '1-URGENT%'
	and o_orderpriority <> '2-HIGH'	and o_orderpriority not like '2-HIGH%'
	and l_shipmode in (':1', ':2')	and (l_shipmode like ':1%' or l_shipmode like ':2%')
	and l_receiptdate >= date ':3'	and l_receiptdate >= cast(':3' as timestamp)
	and l_receiptdate < date ':3' + interval '1' year	and l_receiptdate < date_add(cast(':3' as timestamp) , interval 24 months)
13	count(*) as custdist	cast(c_count as int), cast(count(1) as int) as custdist
	count(o_orderkey)	count(o_orderkey) as c_count
	customer left outer join orders on	customer left outer join orders o on
	c_custkey = o_custkey	c.c_custkey = o.o_custkey
	and o_comment not like '%:1%:2%'	and not o.o_comment like '%:1%:2%'
	) as c_orders (c_custkey, c_count)	) c_orders
14	and l_shipdate >= date ':1'	and l_shipdate >= cast(':1' as timestamp)
	and l_shipdate < date ':1' + interval '1' month;	and l_shipdate < date_add(cast(':1' as timestamp),interval 1 months);
15	create view revenue:s (supplier_no, total_revenue) as	create table revenue:s as
	l_shipdate < date ':1' + interval '3' month	l_shipdate < date_add(cast(':1' as timestamp),interval 3 months)
17	and p_brand = ':1'	and p_brand like ':1%'
	and p_container = ':2'	and p_container like ':2%'
19	sum(l_extendedprice* (1 - l_discount)) as revenue	sum(l_extendedprice* (1 - l_discount)) as revenue
	and l_shipdate >= date ':2'	and l_shipdate >= cast(':2' as timestamp
	and l_shipdate < date ':2' + interval '1' year	and l_shipdate < date_add(cast(':2' as timestamp),interval 24 months)
	and n_name = ':3'	and n_name like ':3%'
21	substring(c_phone from 1 for 2) as cntrycode,	substr(c_phone,1,2) as cntrycode,
	substring(c_phone from 1 for 2) in	substr(c_phone,1,2) in
	and substring(c_phone from 1 for 2) in	and substr(c_phone,1,2) in

alter the performance characteristic of the main query. We also rewrote the *revenue* view from the original as CTAS.

Listing 1: Original Query 15

```
create view revenue:s (supplier_no, total_revenue) as
  select l_suppkey, sum(l_extendedprice * (1 - l_discount))
  from lineitem
  where l_shipdate >= date ':1'
    and l_shipdate < date ':1' + interval '3' month
  group by l_suppkey;
select s_suppkey, s_name,s_address, s_phone, total_revenue
from supplier, revenue:s
where s_suppkey = supplier_no
    and total_revenue = (select max(total_revenue)
                         from revenue:s)
order by s_suppkey;
drop view revenue:s;
```

Listing 2: Modified Query 15

```
create table revenue:s as
select l_suppkey as supplier_no,
       sum(l_extendedprice * (1 - l_discount)) as total_revenue
from lineitem
where l_shipdate >= '1996-01-01' and l_shipdate < '1996-04-01'
group by l_suppkey;
create table temp_table_q15:s as
select max(total_revenue) as max_revenue
from revenue:s;
select s_suppkey, s_name, s_address, s_phone, total_revenue
from supplier join revenue:s on
       s_suppkey = supplier_no
       join max_revenue:s on
      table temp_table_q15=max_revenue
order by s_suppkey;
drop table revenue:s;
drop tabletable temp_table_q15:s;
```

2.2.2 Query 18. The changes to Query 18 are summarized in Listing lst:ModifiedQuery18. Impala was not able to execute the original query (see Listing lst:OriginalQuery18) because of the join condition *o_orderkey IN sub-query*. Similarly to Query 15, the subquery being uncorrelated, we rewrote it as a CTAS.

Listing 3: Original Query 18

```
select c_name, c_custkey, o_orderkey, o_orderdate,
       o_totalprice, sum(l_quantity)
from customer, orders, lineitem
where o_orderkey in (select l_orderkey
                     from lineitem
                     group by l_orderkey
                     having sum(l_quantity) > :10)
  and c_custkey = o_custkey
  and o_orderkey = l_orderkey
group by c_name, c_custkey, o_orderkey,
       o_orderdate, o_totalprice
order by o_totalprice desc, o_orderdate;
```

Listing 4: Modified Query 18

```
create table temp_table_q18_:s as
select l_orderkey, sum(l_quantity) as temp_sum_of_quantity
from lineitem
group by l_orderkey;
select c_name,c_custkey,o_orderkey,o_orderdate,
     cast(o_totalprice*1000 as int)/1000,sum(l_quantity)
from customer join orders on c_custkey = o_custkey
  join temp_table_q18_:s ttq18 on o_orderkey = ttq18.l_orderkey
       and ttq18.temp_sum_of_quantity > :1
  join lineitem l on o.o_orderkey = l.l_orderkey
group by c_name,c_custkey,o_orderkey,
o_orderdate,cast(o_totalprice*1000 as int)
order by cast(o_totalprice*1000 as int) desc,o_orderdate;
drop table temp_table_q18_:s;
```

Listing 5: Original Query 21

```
select s_name, count(*) as numwait
from supplier, lineitem l1, orders, nation
where s_suppkey = l1.l_suppkey
  and o_orderkey = l1.l_orderkey
  and o_orderstatus = 'F'
  and l1.l_receiptdate > l1.l_commitdate
  and exists (select *
              from lineitem l2
              where l2.l_orderkey = l1.l_orderkey
                 and l2.l_suppkey <> l1.l_suppkey)
       and not exists (select *
                 from lineitem l3
                 where l3.l_orderkey = l1.l_orderkey
                 and l3.l_suppkey <> l1.l_suppkey
                 and l3.l_receiptdate > l3.l_commitdate)
       and s_nationkey = n_nationkey
       and n_name = ':1'
group by s_name
order by numwait desc, s_name;
```

2.2.3 Query 21. The changes to Query 21 are summarized in Listing lst:ModifiedQuery21. Impala was not able to execute the original query (see Listing lst:OriginalQuery21) because of the *exists* and *not exists* join conditions. We rewrote the these join conditions as series of outer joins.

Listing 6: Modified Query 21

```
create table temp_table1_q21:s as
select l_orderkey, cast(count(distinct l_suppkey) as int)
     ,max(l_suppkey) as max_suppkey
from lineitem
group by l_orderkey;
create table table temp_table2_q21:s as
  select l_orderkey, cast(count(distinct l_suppkey) as int)
     , max(l_suppkey) as max_suppkey
from lineitem
where l_receiptdate > l_commitdate
group by l_orderkey;
select s_name, cast(count(1) as int) as numwait
from (select s_name from
        (select s_name, t2.l_orderkey, l_suppkey
              ,count_suppkey, max_suppkey
         from table temp_table2_q21:s t2 right outer join
           (select s_name, l_orderkey, l_suppkey
            from (select s_name, t1.l_orderkey, l_suppkey
                  ,count_suppkey, max_suppkey
                  from table temp_table1_q21:s t1 join
                     (select s_name, l_orderkey, l_suppkey
                      from orders o join
                         (select s_name, l_orderkey, l_suppkey
                          from nation n join supplier s on
                               s.s_nationkey = n.n_nationkey
                               and n.n_name like 'SAUDI_ARABIA%'
                               join lineitem l on
                               s.s_suppkey = l.l_suppkey
                          where l.l_receiptdate > l.l_commitdate
                         ) l1 on o.o_orderkey = l1.l_orderkey
                             and o.o_orderstatus = 'F'
                      ) l2 on l2.l_orderkey = t1.l_orderkey
                  ) a
            where (count_suppkey > 1) or ((count_suppkey=1)
              and (l_suppkey <> max_suppkey))
           ) l3 on l3.l_orderkey = t2.l_orderkey
        ) b
    where (count_suppkey is null) or ((count_suppkey=1)
                        and (l_suppkey = max_suppkey))
  )c
group by s_name
order by numwait desc, s_name;
drop table table temp_table1_q21;
drop table table temp_table2_q21;
```

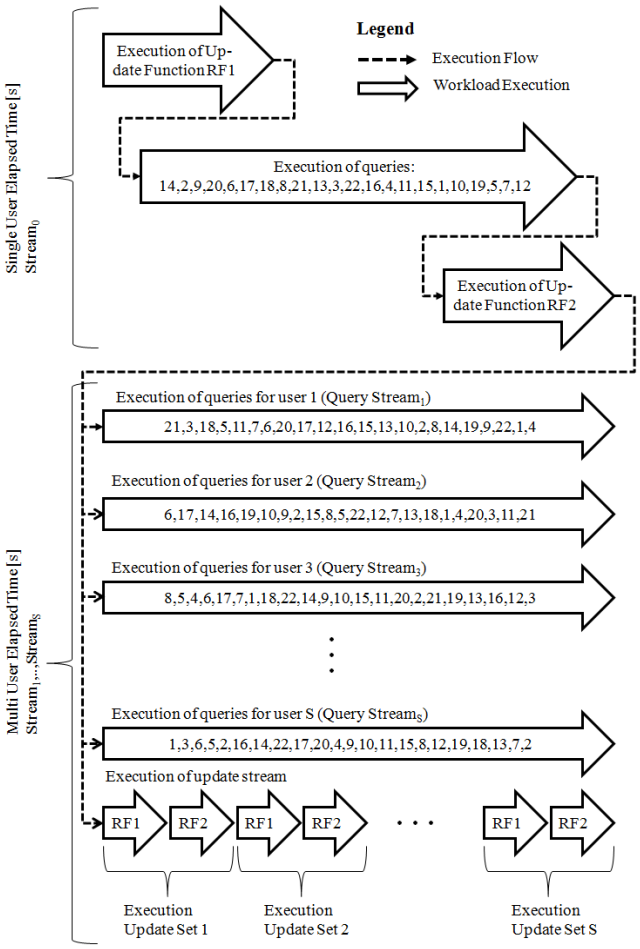

Figure 2: Execution rules of the TPC-H benchmark

2.3 Comparison Metrics

TPC-H defines one main performance metrics, the composite performance, which is calculated as the geometric mean of the single-user metric (*Power@Size*) and the multi-user metric (*Throughput@size*). Size is the scale factor (*SF*), which is a dimensionless quantity. It roughly represents the raw data set size in GB. Power@Size is computed as the geometric mean of the elapsed times for all queries and both refresh functions obtained during the single-user run. Its unit is queries per hour. For a given scale-factor (*SF*), individual query elapsed times (QI_q) $q \in 1..22$ and update functions ($RI_1, RI2$), the single-user metric is computed as:

$$Power@Size = \frac{3600 * SF}{\sqrt[24]{\prod_{q=1}^{22} QI_q \prod_{q=1}^{2} RI_q}} \quad (2)$$

With S being the number of concurrent users during the multi-user run and T_s being the elapsed time multi-user test, the multi-user metric is computed as:

$$Throughput@Size = \frac{3600 * S * 22 * SF}{T_s} \quad (3)$$

The composite performance metric is then calculated as the geometric mean of the single- and multi-user metrics as follows:

$$QphH@Size = \sqrt{Power@Size * Throughput@Size} \quad (4)$$

2.4 Incorporating TPC-Energy into TPC-H

The TPC-Energy specification [13] is designed to augment any TPC benchmark by allowing for the reporting of an energy metric alongside its performance metrics. While a benchmark's performance metrics measures the amount of work completed per unit of time, the TPC-Energy metric measures the energy consumption corresponding to the amount of work completed. TPC-Energy's metric is the ratio of the energy consumed by the entire system *Watt Seconds* $[Ws]$ to work completed (number of transactions, queries, transformations) during the benchmark interval $B_{Interval}$ $[s]$. After moving the time element to the denominator, the TPC-Energy metric is plainly represented as *Watts/Performance*.

TPC benchmarks measure the performance of different types of workloads, some of which are time-based, other are task-based. They use different types of metrics and, because they are technology agnostic, the systems being measured are very diverse in terms of type of system components, architecture and number of tiers.

TPC-C and TPC-E follow a time based benchmark model. They report performance as the transaction throughput during steady state condition. TPC-H, on the other hand, follows a static task benchmark model, which is divided into three distinct measurement t tests: (i) *load test*, (ii) *single-user test*, and (iii) *multi-user test*. All three tests exhibit an oscillating system utilization behavior. To deal with such scenarios, the TPC-Energy specification requires measuring power $P_i[W]$ $i \in 1, 2, ..n$ of the entire system for each interval in addition to the performance measurements T_i, and then independently determining the combined value for power $P[W]$ and performance for all intervals using weights corresponding to the duration of each interval:

$$P = \frac{OverallWork}{OverallEnergy} = \frac{\sum_{i=1}^{n} T_i * S_i}{\sum_{i=1}^{n} P_i * S_i} \quad (5)$$

The primary metric, reported by TPC-Energy, is in the form of *Watts per Performance* for the overall System Under Test (SUT) [1] where the performance units are particular to each TPC Benchmark. For TPC-H it would be Watts/QphH. The energy consumption is measured for all subsystems active for the duration of the benchmark run. This includes servers, storage, clients, network switches. The TPC-Energy Specification also defines optional secondary metrics. The purpose of these secondary metrics is to allow more detailed comparisons and analysis of the result for system components such as server chassis, storage system, network gear etc. The secondary metrics are represented in similar units as the primary metric, that is, Watts/Performance, and the summation of all individual secondary metrics equals the primary metric. This is because both the primary and secondary metrics share a common value for the denominator —the performance value. This was done by design when developing the benchmark specification to allow end-users to see the contribution of the subsystems (represented by

[1]The system under test is a TPC defined term for system being tested by a benchmark

the secondary metrics) to the overall system results (represented by the primary metric).

In addition to these primary and secondary metrics, the TPC-Energy specification also calls for reporting the *idle power*, which is defined as the energy consumption of the SUT within 30 minutes of the completion of the benchmark run. The intent is to represent the amount of energy consumption of a measured system in a state "ready to accept work". This is useful to customers who have systems that have periods of idle but require the system to respond to a request for work at any time.

In order to enable the reporting of energy numbers in TPC-H the following three clauses need to be amended:

(1) Clause 0.1 has to be amended with the following wording: *To be compliant with the optional TPC-Energy standard, the additional primary metric, expressed as watts-per-QphH, must be reported. The requirements of the TPC-Energy Specification can be found at www.tpc.org.*
(2) Clause 5.4 has to be amended with the following wording: *When the optional TPC-Energy standard is used, the additional primary metric, expressed as* $\frac{watts}{QphH}$*, must be reported. In addition, the requirements of the TPC-Energy Specification, located at www.tpc.org, must be met.*
(3) Clause 8.3.7 must be amended with *When the optional TPC-Energy standard is used, the additional requirements and formatting of TPC-Energy related items in the executive summary must be reported and used. In addition, the requirements of the TPC- Energy Specification, located at www.tpc.org, must be met.*

3 EXPERIMENTAL SETUP

For our experiments we use six HUAWEI Tecal RH2288 V2 Rack servers, each with 2 Intel Xeon Processor E5-2680 (Sandy Bridge-EP) running at 2.7 GHz. Each processor has 8 cores (16 hyperthreaded) with an 20MB L3 cache. They are connected through two QuickPath (QPI) links, each providing a unidirectional transmission rate of up to 8.0 GB/s. Each server has 24 8GB double data rate 3 (DDR3) at 1066Mhz dimms of main memory with a total ot 192GB. Each server is configured with eight 2.5" SAS HDDs with 7.2TB capacity. One SAS disk hosts the OS and the remaining 7 are configured for HDFS. The server provides four onboard gigabit Ethernet (GE) ports, of which two were bonded to doube bandwidth.

Our system runs Cloudera's CDH 5 on CentOS 6.5. It is configured as one master node and five worker nodes. Each of the nodes has 189.1 GBytes of memory.

4 POWER MEASUREMENT METHODS

This sections gives a brief overview of the measurement methods we are using: (i) Analytical power consumption model (ii) IPMI (iii) Power meter

4.1 Analytical Power Consumption Model

The analytical power consumption models for online transaction processing and analytical workloads, presented in [7], are based on the assumption that the peak power consumption of an entire system can be derived from the aggregate of the nameplate power consumptions of its individual components [2]. Each model

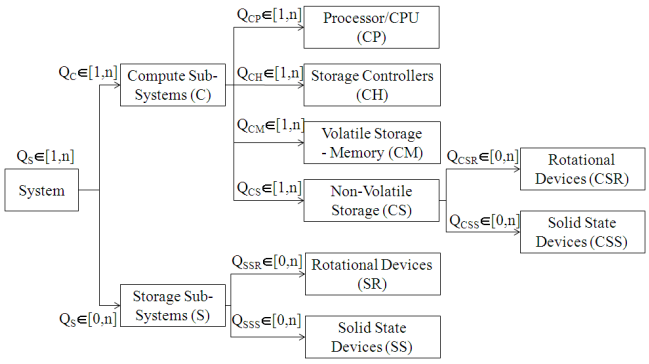

Figure 3: Generalized power consumption model: source [7]

follows the same general approach: The nameplate power information of major system components (see Table 2), such as processor (CPU), volatile storage, internal non-volatile storage devices, that is, rotational disks and solid state memory, and external storage sub-systems, that is, enclosures with non-volatile storage devices, are aggregated discounting the nameplate overhead. Additional power of supporting components, such as motherboards and fans, is calculated with a combination of a fixed overhead and a percentage of the power consumption of the components they support. The models do not account for the power necessary for the air conditioning systems of data centers.

Table 2: Major power consuming components

#	Component	Description
1	Power Supply	2 hot-swappable PSUs
2	Main Board	GIGABYTE GA-EXX58-UD3R
3	CPU	Intel Core i7 920 LA1366
4	Main Memory	24 DDR3 8GB PC3-10600
5	HDD	Seagate, ST91000640NS
4	CPU FAN	4-hot-swappable

Figure 3 shows the hierarchy of the major system components included in our power estimation models. Each component in this hierarchy is abbreviated with up to three capital letters, as indicated in parenthesis. TPC systems may consist of two types of sub-systems, namely compute sub-systems (*C*) and storage sub-systems (*S*). In case of a clustered system and systems that have multi-tier architectures, there can be multiple compute sub-systems. We also refer to the compute sub-systems as servers. Each server consists of one or more compute units (*CP*), that is, processors/CPUs, a number of storage controllers to connect to external storage enclosures (*CH*), some sort of volatile storage, usually memory DRAM DIMMs (*CM*), some non-volatile memory (*CS*), traditionally rotational devices (*CSR*), but recently also Solid State Devices (*CSS*) and supporting components, such as the main board and cooling fans. We also refer to the supporting devices of servers as chassis. The storage sub-system (*S*), which is used to store data persistently, consists of non- volatile storage devices, traditionally rotational

devices (*SR*), but recently also Solid State Devices (*SS*). We also refer to the storage sub-systems as storage enclosures.

Each of these components may occur multiple times in a system and each occurrence may have different nameplate characteristics. Hence, we enumerate them with an index on each level. For instance, the second CPU in the first compute sub-system is labeled $CP_{1,2}$. The 5th rotational device in the second supporting component of the first storage sub-system is labeled $SSR_{1,2,5}$. We refer to the quantities and power consumptions of these components with Q and P respectively. For example, the number of CPUs in the first compute sub-system is $Q(CP)$ and the power consumption of the second CPU in the first compute sub-system is $P(CP_{1,2})$.

4.2 Power Meter Setup

For the power consumption measurements of our system we use the industry standard power meter HIOKI 3360. It is a multi-channel meter supporting clamps and voltage probes to measure power on single- to three-phase lines. The 33 Family of power meters is approved by TPC to be used in energy measurements for TPC benchmarks. Power of the entire System was measured.

The Intel CPU is packaged in Intel's LGA775 socket on the main board. We measure the CPU input current and voltage at its 8-pin power plug. The CPU is powered with one 4 pin fan-connector, its measurement is relatively straightforward. A GPU card is plugged into a PCI-Express slot on the main board, it is mainly powered by +12V and +3.3V power from PCI- Express pins, and an additional +12V power directly from the PSU. We measure the current through auxiliary power line with a clamp probe, and measure the PCI power at the main board power inputs. The memory power consumption is dependent on the memory usage of the software running. We make an approximation by measuring power changes on the main board. We use National Instruments USB-6216 BNC data acquisition, Fluke i30s / i310s current probes, and Yokogawa 700925 voltage probe. The room temperature was kept constant at 23 degrees Celsius ± 0.5 degrees Celsius. We use LabView 8.5 as oscilloscopes and analyzer for the results data analysis. By testing the power responses of each component involved in a sample matrix multiplication, we record the real time voltage and current from measurement readings. The product of voltage and current is the instant power at each sampling point during measurement. We aggregate all data into a single power consumption number [W] and plot that data over time.

4.3 Intelligent Platform Management Interface

The Intelligent Platform Management Interface (IPMI) provides management and monitoring capabilities independent of its host system's CPU, firmware (FM) and operating system (OS). It is a powerful framework that has been initiated by Intel in 1998 and has since been adopted by many other system vendors to manage their systems, including power management.

Motherboards that support IPMI 2.0 [4] are able to monitor the power usage of various system components using a baseboard management controller (BMC) and power sensors. The BMC is a specialized microcontroller embedded on the motherboard. It manages the interface between the system management software and the platform hardware.

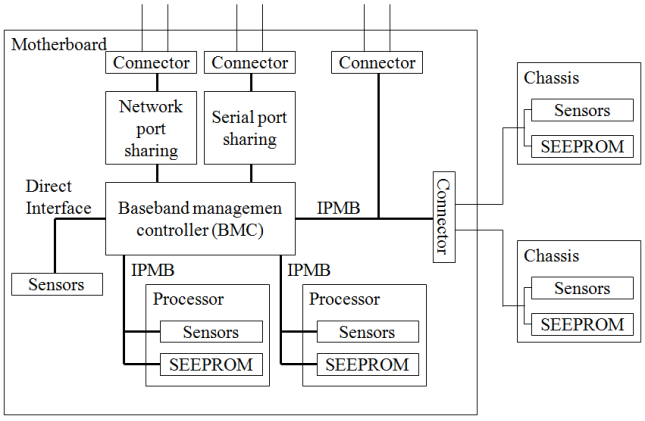

Figure 4: Architecture of IPMI and BMC

In general an IPMI system consists of one BMC and many satellite controllers that are distributed among different system modules, as shown in Figure 4. The satellite controllers within the same chassis connect to the BMC through the IPMI bus or bridge.

The system can be managed with the Remote Management Control Protocol (RMCP). The sensor data record (SDR) repository stores the readings of the individual sensors on the board, which can be temperatures, fan speeds, and voltages.

5 EXPERIMENTAL RESULTS

This section summarizes the power consumption quantities obtained with the three power measurement methods. Firstly, we apply the analytical power consumption model to our hardware setup. The analytical model, being the most conservative, will give us an estimate for the upper bound of the amount of power our system will use during the two types of tests:(i) single-user test, and (ii) multi-user-test. Secondly, we obtain actual power measurements using IPMI and power meters.

5.1 Results Using the Analytical Power Consumption Model

We apply the analytical model to the compute units, the volatile and non-volatile storage units and the supporting components of the compute sub-system.

5.1.1 Power Consumption of Compute Unit. Our system has no separate storage sub-system, hence we only need to consider the compute subsystems without the storage controller, host bus adaptor (HBA), which are usualy Peripheral Component Interconnect (PCI) cards. We obtain the peak power consumption of the compute units (processors/CPUs), which are usually specified as thermal design power (TDP) from the Intel specification [3]. According to it the Intel Xeon E5-2600 series processors's TDP is 130W. Hence, the power consumption of both compute units on each compute sub-system $i \in 1..6$ is:

$$P(CP_i) = 2 * 130W = 260W \quad (6)$$

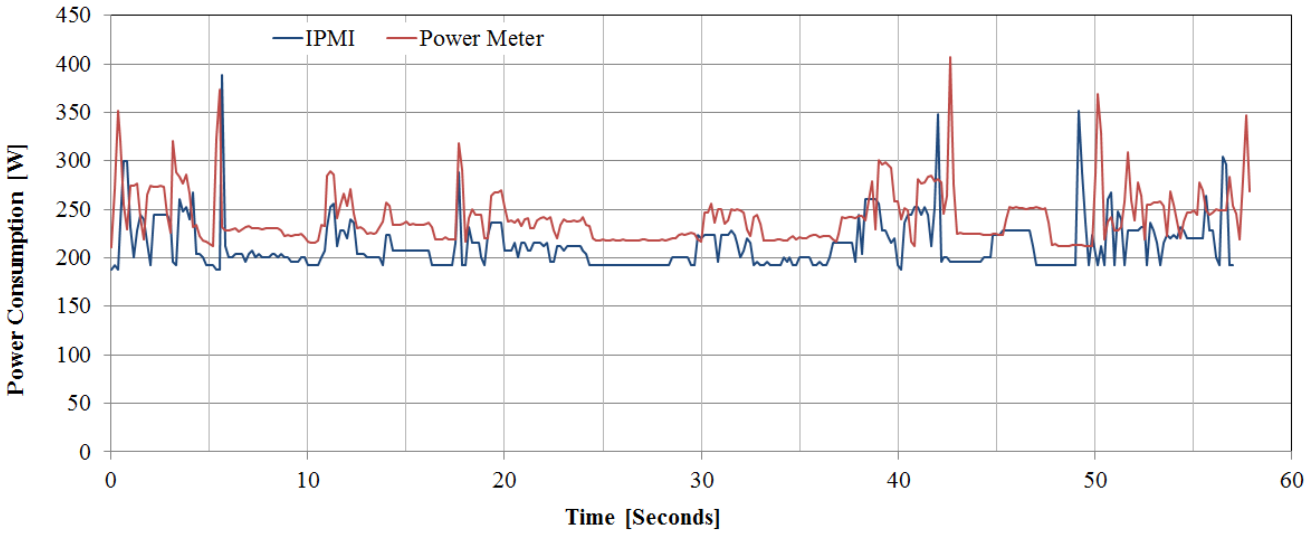

Figure 5: IPMI and power meter readings during the execution of a TPC-H single-user run

5.1.2 Power Consumption of Volatile Storage. Similarly to the compute node nameplate power consumption, we obtain the peak power consumption of our volatile storage (DRAM memory DIMMs) from the manufacturer's website. The power consumption of the entire volatile storage in one compute sub-system i ($i \in 1..8$) can, therefore, be calculated as:

$$P(CM_i) = 24 * 5.21W = 125.0W \quad (7)$$

5.1.3 Power Consumption of Non-Volatile Storage. Each of our compute sub-system contains only rotational storage devices (CSR), that is, disk drives. Peak power consumption levels of disk drives vary widely with the disk's form factor (FF), size, and rotational speed. FF refers to the form factor of the drive. Each of our compute sub-systems holds eight disk drives, each with a peak power consumption of 14.2W. The power consumption of the entire non-volatile storage in each compute sub-system i ($i \in 1..8$) can be estimated with:

$$P(CSR_i) = 8 * 14.2W = 113.6W \quad (8)$$

5.1.4 Power Consumption of Compute Sub-System (servers). In addition to compute units, volatile and non-volatile memory, we need to add the power consumption of the supporting components of the compute sub-system to estimate their total power consumption. Supporting components are the main board, cooling fans, caches, etc. They are also referred to as the server chassis. Studies [7] and [6] suggest that the power consumption of the server chassis can be expressed as a percentage (30%) of the nameplate power consumption of its main components plus a fixed overhead (100W). Hence, we compute the power consumption of one of our compute sub-system $i \in 1..6$ as:

$$P(C_i) = (P(CP_i) + P(CM_i) + P(CSR_i)) * 1.3 + 100 \quad (9)$$

$$P(C_i) = 498.6 * 1.3 + 100 = 748.2W \quad (10)$$

5.1.5 Power Consumption of the Entire System. The power consumption of the entire system is the aggregate of the power consumption of all six nodes: $6 * 748.2 = 4489.1W$. Using the elapsed times for the single- and multi- user runs of 58.3s and 208.8s respectively, the system wide power consumption of our system using the nameplate power consumption estimation model during the single-user run is is $58.3 * 748.2W = 43,620.1Ws = 12.2kWh$. During the multi-user run the system wide power consumption of our system is $208.8 * 748.2W = 156224.2Ws = 43.4kWh$. The power consumption of our system during the entire measured interval as it would be used during a TPC-H publication is $(58.3 + 208.8) * 748.2W = 199,844.2Ws = 55.6kWh$. It is understood that using nameplate power consumption is a very conservative estimate.

5.2 IPMI and Power Meter Measurement Results

With the power consumption estimate from our analytical model we have established an upper bound for the total power consumption. Results from the IPMI and the power meter setups will be more accurate. The following sections summarize results obtained from IPMI and actual power meters. Firstly, we present the results from the single-user run.

5.2.1 Single-User Power Measurement. Figure 5 plots the power consumption of the entire system as measured by IPMI and power meters during a single-user run of TPC-H. We obtained both the IPMI and power meter power consumption readings from the same single-user run. IPMI reports a minimum power consumption of 188.0 W, a maximum power consumption of 388.1 W, and an average power consumption of 241.3 W. The total power consumption during the measurement interval is $12172.0Ws$=$3.4kWh$.

The power meter reports a minimum power consumption of 211.5 W, a maximum power consumption of 407.1 W and an average power consumption of 241.7 W. The total power consumption

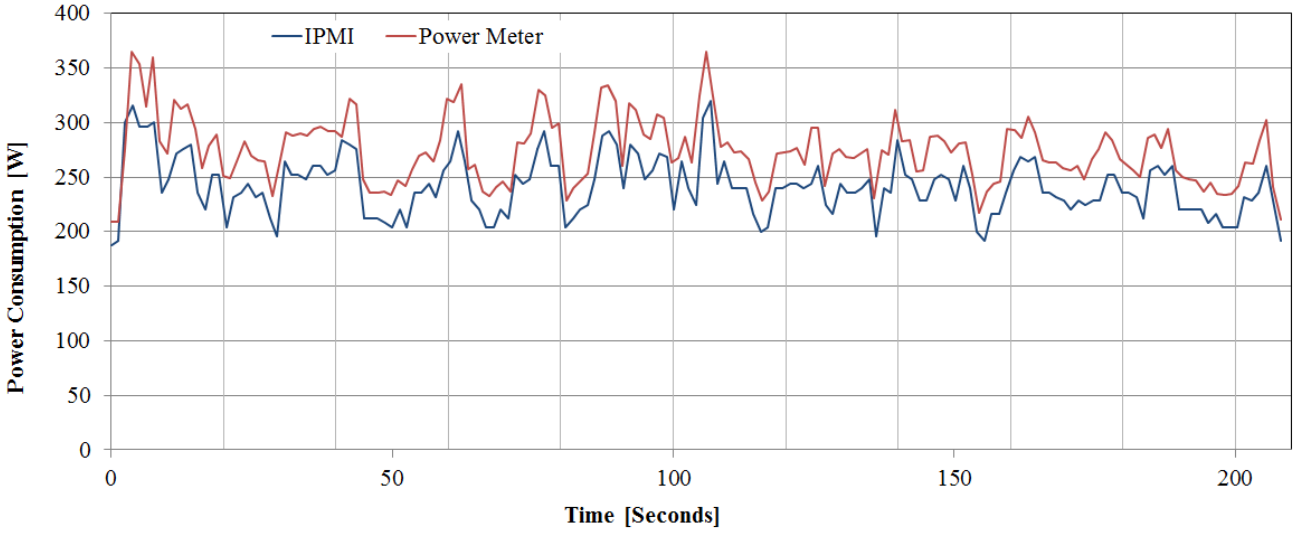

Figure 6: IPMI and power meter readings readings during the execution of a TPC-H multi-user run

during the measurement interval is 13999.8Ws=3.9kWh. On average the power meter reports measurement that are 11.7% higher compared to the numbers reported by the IPMI tool. The total power consumption reported by IPMI is 72% lower compared to the estimate of the analytical model and the total power consumption reported by the power meter is 68% compared to the estimate of the analytical model.

The graphs in Figure 5 show that the IPMI readings follow the power meter readings. 30% of the IPMI readings are within 10% of the power meter readings and 88% of the IPMI readings are within 20% of the power meter readings. There are, however, some outliers that are -65% and +52% off from the power meter readings.

5.2.2 Multi-User Power Measurement. Figure 6 shows the power consumption of the entire system as measured by IPMI and power meters during a multi-user run of TPC-H (four concurrent user). As in the single-user case we obtained both the IPMI and power meter power consumption readings from the same multi-user run. IPMI reports a minimum power consumption of 188.0 W, a maximum power consumption of 320.0 W, and an average power consumption of 241.3 W. The total power consumption during the measurement interval is 50264.1Ws=14.0kWh.

The power meter reports a minimum power consumption of 208.9 W, a maximum power consumption of 365.0 W and an average power consumption of 274.2 W. The total power consumption during the multi-user run is 55493.0Ws=15.4kWh. During the multi-user run the power consumption numbers reported by the power meter 9.4% higher compared to the numbers reported by the IPMI tool. This is slightly lower than during the single-user run (11.7%). The total power consumption during the multi-user run as reported by IPMI is 68% lower compared to the estimate of the analytical model and the total power consumption reported by the power meter is 64% compared to the estimate of the analytical model. These numbers are slightly lower compared to the single-user run (72% and 68%).

Similar to the single-user run the IPMI readings follow the power meter readings very closely. 37% of the IPMI readings are within 10% of the power meter readings and 78% are within 20%. Contrary to the single-user run the outliers in the IPMI readings are smaller. They vary between -20% and +30%.

Figures 7 and 8 graph the percent difference between the IPMI and power meter reading for each sample taken during the single- and multi-user runs. The graphs display the difference in ascending order to show the distribution of the differences.

6 CONCLUSION

With rising energy costs and increasing power consumption due to the ever-growing demand for compute power (servers, storage, networks), energy efficiency is a top priory for system vendors and customers. Measuring energy in a benchmark environment can be very laborious and complicated if the system is very large, which TPC benchmark systems tend to be. As of today, there have been only three TPC benchmark results published that report energy consumption non of which are TPC-H results . In this paper, we have argued that the low number of TPC publications is due to the large setups required in TPC benchmarks and the, subsequently, complicated measurement setup.

We have amended TPC-H to include power measurements and have analyzed three approaches in measuring energy consumption: (i) power meters, (ii) IPMI and (iii) analytical power consumption model using nameplate power consumption.

Analytical power consumption mode using nameplate power consumption delivers an upper bound for energy consumption and is not very accurate. However, our results show that IPMI provides an alternative to gathering power consumption numbers on a system, especially because IPMI is readily available and can be configured very easily.

The following table summarizes the results of the three different quantitative approaches, discussed in this paper.

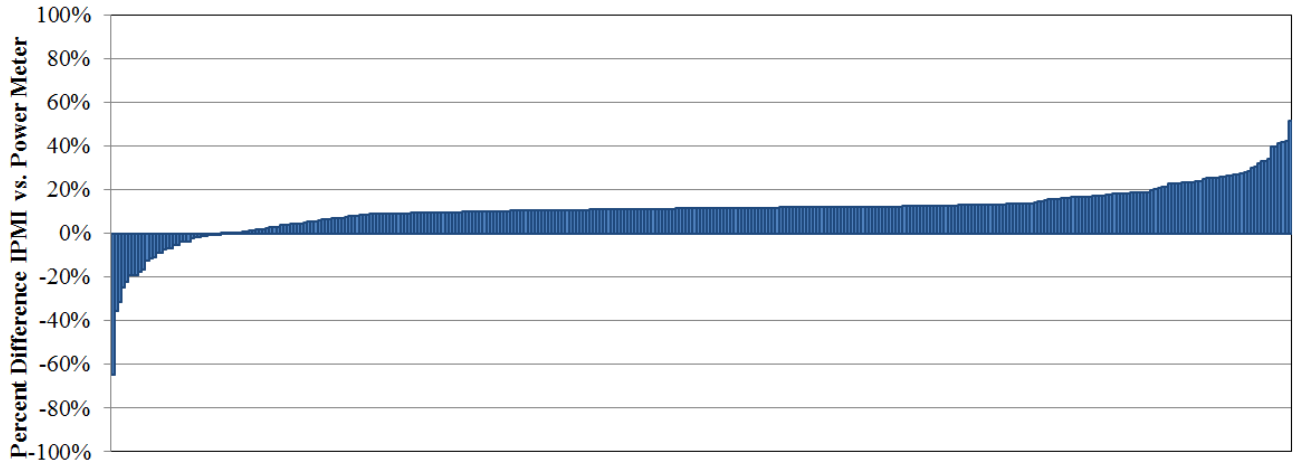

Figure 7: Sorted percent differences of IPMI vs. power meters readings during a single-user run

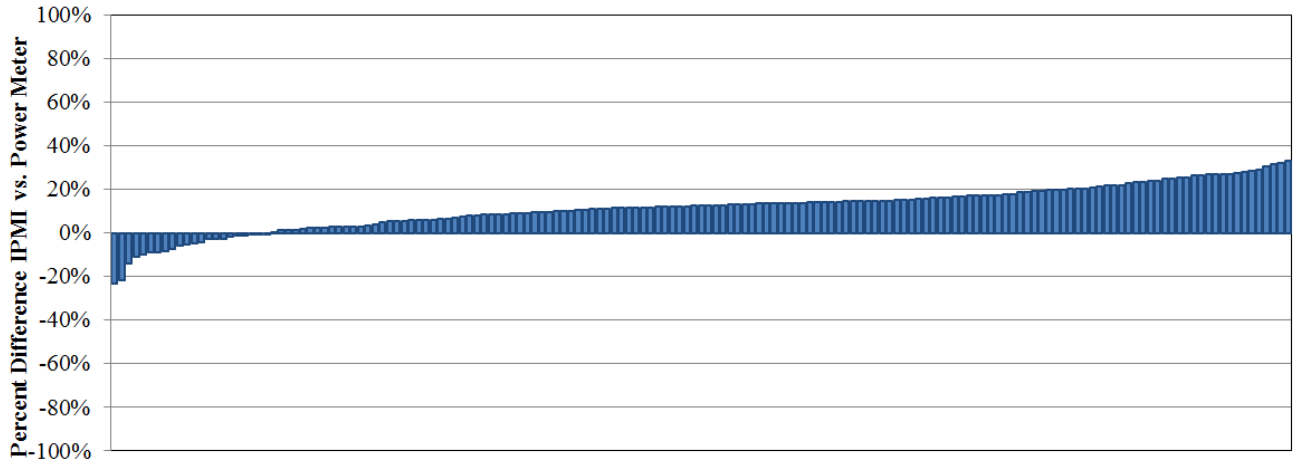

Figure 8: Sorted percent differences of IPMI vs. power meters readings during a multi-user run

Table 3: Power readings using different methods during TPC-H Runs

Test	Nameplate	Power Meter	IPMI
Single-User	12.2 kWh	3.4 kWh	3.9 kWh
Multi-User	43.4 kWh	14.0 kWh	15.4 kWh

The nameplate power consumption model overestimates the power consumption by more than 3x, while the difference between the IPMI and the external power meters is about 10%. It is up to benchmark consortia to allow the use of IPMI as an alternative to traditional power meters in order to increase adaptability of IPMI. It is clearly a trade off between effort/money and accuracy.

ACKNOWLEDGMENTS

This work was partially supported by the German Ministry for Education and Research as BBDC (01IS14013A).

REFERENCES

[1] C. Ballinger. TPC-D: benchmarking for decision support. In *The Benchmark Handbook for Database and Transaction Systems (2nd Edition)*. 1993.

[2] X. Fan, W. Weber, and L. A. Barroso. Power provisioning for a warehouse-sized computer. In D. M. Tullsen and B. Calder, editors, *34th International Symposium on Computer Architecture (ISCA 2007), June 9-13, 2007, San Diego, California, USA*, pages 13–23. ACM, 2007.

[3] Intel. Intel Xeon Processor E5-2680 (20M Cache, 2.70 GHz, 8.00 GT/s Intel QPI), 2013.
[4] Intel. IPMI Specificationv2.0, rev. 1.1: Document, 2013.
[5] M. Poess and C. Floyd. New TPC Benchmarks for Decision Support and Web Commerce. *SIGMOD Record*, 29(4):64–71, 2000.
[6] M. Poess and R. O. Nambiar. A power consumption analysis of decision support systems. In A. Adamson, A. B. Bondi, C. Juiz, and M. S. Squillante, editors, *Proceedings of the first joint WOSP/SIPEW International Conference on Performance Engineering, San Jose, California, USA, January 28-30, 2010*, pages 147–152. ACM, 2010.
[7] M. Poess and R. O. Nambiar. Power based performance and capacity estimation models for enterprise information systems. *IEEE Data Eng. Bull.*, 34(1):34–49, 2011.
[8] M. Poess, R. O. Nambiar, K. Vaid, J. M. Stephens, K. Huppler, and E. Haines. Energy benchmarks: a detailed analysis. In H. de Meer, S. Singh, and T. Braun, editors, *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking, e-Energy 2010, Passau, Germany, April 13-15, 2010*, pages 131–140. ACM, 2010.
[9] SPC. Storage Performance Council. http://www.storageperformance.org/home/.
[10] SPEC. The Software Performenance Evalution Corporation (SPEC). http://www.businesswire.com/news/home/20070417005047/en/SAS-Smashes-ETL-World-Record-Establishing-New.
[11] SPEC. SPECpower ssj 2008 Specification. https://www.spec.org/power_ssj2008/, 2017.
[12] TPC. Home Page TPC. http://www.tpc.org, 2017.
[13] TPC. The TPC Energy Specification. http://www.tpc.org/TPC_Documents_Current_Versions/pdf/TPC-Energy_v1.5.0.pdf, 2017.