Shall I Use Heterogeneous Data Centers? – A Case Study on Video on Demand Systems

Shi Feng*, Hong Zhang* and Wenguang Chen†

* *Department of Computer Science and Technology*
Tsinghua National Laboratory for Information Science and Technology
Beijing, 100084, China
Email: {fredfsh,zhanghong1009}@gmail.com
† *Research Institute of Tsinghua University in Shenzhen*
Shenzhen, 518057, China
Email: cwg@tsinghua.edu.cn

***Abstract*—Recent research work and industrial experience show that heterogeneous data centers can be more cost-effective for some applications, especially those with intensive I/O operations. However, for a given application, it remains difficult to decide whether the application should be deployed on homogeneous nodes or heterogeneous nodes.**

In this paper, we perform a case study on finding optimized configurations for Video on Demand(VOD) systems. We examine two major modules in VOD systems, the streaming module and the transcoding module. In this study, we measure performance and power on different configurations by varying these factors: processor type, disk type, number of disks, operating systems and module mapping modes.

We find that hosting VOD systems on heterogeneous nodes reduces no more than 35% power or cost per unit work done, compared with the server-only configuration. In Windows, multiplexing the two modules on servers achieves comparable power and cost reduction, but it is even worse than the non-multiplexing solution in Linux. The anti-intuitive and non-consistent results reflect the intrinsic complexity of finding an optimized configuration for a VOD system. Empirical data are necessary to support quantitative instead of qualitative analysis.

***Keywords*-power, heterogeneity, data center, video-on-demand**

I. Introduction

Wimpy nodes (e.g. Atom based servers) are proved to be power and cost effective, when equipped with solid-state-drives (SSDs), for some data-intensive applications like key-value stores [1], [2], but not suitable to handle complex workload like database applications [3], [4]. It is important to answer the following question: should a given application be deployed on servers, wimpy nodes, or a mixed cluster of both?

Furthermore, an application may have multiple modules, each with different characteristics. Should we deploy different modules to homogeneous nodes or heterogeneous nodes? Or should we multiplex different modules on the same machine?

In this paper we perform a case study on VOD systems. Previous work suggests power and cost efficiency of a platform should be determined by actually running the workload on the platform [4], [5]. We conduct extensive experiments to find the optimized configuration for VOD systems. We use Helix Server with Helix Producer in Windows, and choose VLC+FFmpeg in Linux. We measure the power and cost of platform combinations over three types of processors and three kinds of disks. Each VOD system contains two modules. We explore the effectiveness of mapping distinct modules to a) distinct nodes of same type; b) different types of nodes and c) a single node.

In a VOD system, the streaming module is I/O-intensive while the transcoding module is CPU-bound. It is intuitive that deploying the streaming module to wimpy nodes with SSDs and the transcoding module to servers could reduce power draw and overall cost. As the two modules stress different components of a computer, it is also intuitive to multiplex them on the same machine.

Our experiment results illustrate how intuition sometimes lies. Deploying the streaming module to wimpy nodes with SSDs and the transcoding module to servers reduces no more than 35% power or cost per unit work done, compared with the server-only solution, in both Windows and Linux. The power and cost efficiency of multiplexing modules on servers shows non-consistent results across different OSes. In Windows, it is as efficient as the heterogeneous approach (less than 10% difference in terms of power performance ratio and cost performance ration). In Linux, such multiplexing mode is even worse than traditional non-multiplexing solution. In Linux, using wimpy nodes with SSDs, for both modules, is the second best configuration among our choices, while in Windows it is the poorest.

These anti-intuitive and non-consistent results imply the intrinsic complexity of the problem. The power and cost efficiency is affected by how well the underlying hardware matches applications, how well the application cooperate

with the operating system, and how well the modules inside the application interact with each other. Finding an optimized hardware configuration for a given application is non-trivial without quantitative empirical results.

Contributions of this paper include: (1) We exhibit a case study on finding the optimized configuration for VOD systems. We measure performance and power with experiments, and calculate power and cost efficiency with carefully constructed models. The whole process can be applied to other Internet applications. (2) We show it is not trivial to find the optimized configuration for a given application, which requires extensive experiments to support quantitative analysis. (3) To our best knowledge, this is the first publication which shows empirical benchmarking results of *multiplexing* different modules of VOD systems on same nodes. All the data in our experiment would be available online at a later time.

The rest of this paper begins with a brief introduction on background knowledge in Section II. Section III describes how we quantify the performance and how we model the power and cost efficiency. Section IV details specifications about the software and hardware we use in the experiment. Section V exhibits experiment result and its implications. Related work can be found in Section VI. The last section draws conclusions.

II. Background

A. Video-on-Demand Systems

In this paper we focus on two major modules of a VOD system: transcoding module and video streaming module. In a production environment there may be other modules like advertising module, recommendation module, etc. We consider only transcoding module and streaming module for two reasons. First, they are necessary while the other modules are not. Second, distinct characteristics of the two modules already exhibit the complexity in our choice of the underlying platform.

A transcoding module encodes raw input, or transcodes videos from one format to another. It performs intensive computation and relatively few I/O operations. A streaming module serves user requests and transfers videos to media playing clients. This part of the VOD system requires high IOPS and bandwidth to serve concurrent video requests.

When a new video is sourced (e.g. uploaded by a user), it is transcoded and preserved in storage once and for all. Each time a user requests for a video, the transcoded version is streamed to the client. Frequently accessed videos may be cached to reduce the response time.

B. Wimpy Nodes with SSDs

Wimpy nodes are platforms with CPUs of less computing power [1], [6], [3]. Compared with powerful processors (e.g. Xeon), wimpy processors (e.g. ARM-based, Atom) feature low power draw and low dollar cost. For example, an Intel Atom N570 has a thermal design power of 8.5W [7]. The value of an Intel Xeon E5620 is 80W [8], one order of magnitude more than the Atom processor.

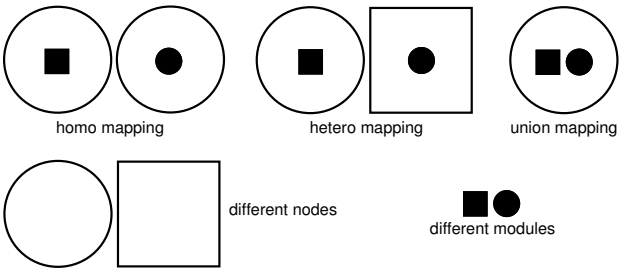


Figure 1: Three modes of module mapping.

Solid-state-drives (SSDs) outperform traditional harddisk drives (HDDs) with low random read latency (hundreds of microseconds compared to HDD's several to tens of milliseconds), and low power consumption. The defects of SSDs are poor write performance and limited endurance (about five years under intensive accesses [9]). Previous research demonstrates that wimpy nodes with SSDs are energy-efficient on some kinds of data-intensive jobs like key-value stores and sorting [1], [6], [2], and are not good at handling some complex jobs like database applications [3], [4].

C. Module Mapping Modes

In a heterogeneous data center, there are nodes in different types. For example, there may be Xeon servers and Atom based servers. Applications like VOD systems may contain multiple modules. There would be many choices on how to map modules of an application to available nodes in a data center. We classify them into three *module mapping modes*, as is depicted in Fig. 1. In the **homo**(geneous) mode, different modules are mapped to different instances of the same type of node. In the **hetero**(geneous) mode, different modules are deployed on nodes of different types. In the **union** mode, different modules are multiplexed on a single node.

Intuitively, the streaming module of a VOD system is I/O-bound and performs well on wimpy nodes with SSDs. The transcoding module is CPU-bound and should be deployed on powerful servers. Furthermore, as the two modules stress I/O components and processors respectively, multiplexing the two modules on a server with SSDs may be a better choice. In this paper we deploy VOD systems on combinations of various processors and disks, in different mapping modes. The whole process gives some insights on finding an optimized platform configuration for a given application like VOD systems. The results of our experiment imply that actually running the application on each type of node is critical in finding the optimized configuration.

III. Evaluation Metrics

Our experiment spans three dimensions in configuration. First, we choose two VOD applications, one in Windows and one in Linux. Second, the hardware platforms are combinations of three types of processors and three types of disks. The number of disks is also varied. Third, we use three module mapping modes. We use **power performance ratio** and **cost performance ratio** to quantify the effectiveness. We run each module of the VOD system on each hardware platform. Our experiment also includes running the two modules simultaneously on the same node. During the run we measure the power and performance, and calculate the cost. Given these data we may derive the above two ratios of running a module on a specific platform. To compare between the mapping modes, we set up a universal Service Level Agreement(SLA) and calculate the power and cost on achieving that SLA, in each mapping mode. This section explains how we quantify the performance, and describes the power and cost models. Details about the software and hardware we use can be found in Section IV.

A. Performance Metrics

A typical SLA may require the VOD service provider handle X concurrent streaming requests while transcode Y hours of videos per second. We quantify the performance as *the capability of finishing the SLA*.

Performance of the streaming module on a platform is quantified as the number of simultaneous streaming requests it supports. We assume the quality-of-service (QoS) requires no less than 95% videos being streamed are *good* playbacks. The playback of a video is deemed good if the actual playback length differs by 5% or less from the length of original video. The videos being streamed are identical copies of a 15-minute-long video.

Performance of the transcoding module on a platform is quantified as the number of hours of videos completely transcoded per second. It is possible that multi-core processors may transcode multiple videos in parallel. The performance is measured as the sum of work done by all transcoding processes. The videos being transcoded are also identical copies of the same video file.

B. Single-Noded Power Model

Power draw is monitored by connecting the node to the AC outlet of a Yokogawa WT210 digital power meter. The device is accepted by Standard Performance Evaluation Corporation (SPEC) used for power efficiency benchmarking [10]. The power consumption is averaged over the test period where power draw is nearly constant. Typically this is within the interval of timespan between beginning of the last instance (streaming or transcoding process) and end of the earliest.

Power performance ratio is defined by

$$p_w = \frac{P_w}{M_w}. \tag{1}$$

P is the power draw and M is the measured performance. The subscript w is either s or t indicating the streaming or transcoding module respectively.

A node consumes power even when it is idle. The optimal working state lies somewhere between it is idle and it is at full run. At that state it achieves minimum power performance ratio. In other words, it consumes minimum power per unit work done. We assume the curve is unimodal. We adopt different strategies to find the optimal working state for the two modules. For the streaming module, we use up to twenty client machines to initiate streaming requests to the streaming server. We first binary-search the maximum number of requests the server may handle. Then we reduce the load in a fine-grained manner and stop when we encounter the first local minimum of the power performance ratio. As the streaming task is I/O-bound, we also watch the effect of disk numbers. We repeat the above process, each time adding one disks, until there is no available slots, or no decrease in the power performance ratio. For the transcoding module, each time we add one more transcoding process and stop when the first local minimum of power performance ratio appears. It is unlikely to go far beyond the number of physical cores of the processor.

C. Single-Noded Cost Model

Our cost model consists of nodes acquisition cost and power cost. A previous study shows that these two components contribute to 60% cost in a data center [11]. Another 25% goes to power distribution and cooling infrastructure. The cost performance ratio is calculated according to

$$c_w = \frac{1}{M_w}(P_w C_e + \frac{C_{hd}}{T}) = p_w C_e + \frac{C_{hd}}{M_w T}. \tag{2}$$

C_e is electricity price, C_{hd} is the dollar cost the node and T is the expected lifetime of hardwares.

In a heterogeneous data center, the cost of job scheduling and facility acquisition is expected to be higher. Wimpy nodes may be less stable and more easily worn-out than high-end servers. A more accurate model may take this maintenance cost, and possible other cost, into account. This is a task for future research. In this paper, modeling the cost is just an attempt towards simulating the real world cases within a heterogeneous data center. However, as the power is measured directly, in can be referenced with more confidence.

D. Scaling the Models

We set up a universal SLA to compare the effectiveness of different mapping modes. The SLA is an ordered pair of integers (X_s, X_t). It indicates the VOD system is able to stream X_s videos while at the same time to transcode

X_t hours of videos per second. The power and cost models above go for running a single module on individual nodes. In this subsection, we explain how to calculate the power and cost of achieving the target SLA in each mapping mode.

Our analysis is based on two assumptions: 1) power (cost) scales linearly with the number of nodes, and 2) number of required nodes scales linearly with workloads. Previous work demonstrates that these are not the cases for database workloads, due to heavy communications [4]. Anyway, scaling proportionally is probably appropriate for VOD systems because neither of the modules requires heavy inter-process communications. Hardly any inter-node communications are involved either. This makes it reasonable to assume power and cost goes approximately linearly with the number of nodes involved, which is further proportional to the workload.

Note that above assumptions never admit proportional power within a single node. The presence of idle power makes this almost impossible. However, as long as each node is utilized to the same extent, we can safely calculate the power (cost) of achieving the target SLA by multiplying the workload with the single-noded power (cost) performance ratio.

Given above assumptions and analysis, the power and cost of achieving the universal SLA, in either homo or hetero mapping mode, are calculated by

$$\begin{aligned} P_{homo/hetero} &= p_s X_s + p_t X_t \\ C_{homo/hetero} &= c_s X_s + c_t X_t. \end{aligned} \tag{3}$$

In the homo mode, p_s and p_t are chosen such that they correspond to the same platform. In the hetero mode, we select the minimum p_s (p_t) among all power (cost) performance ratios. The idea is trivial: we choose the best platform for each module. The hetero mode must be no worse than the home mode in power and cost efficiency. Our experiment result quantifies the gap between these two modes.

In the union mapping mode, the case is more complicated. When there are different number of transcoding processes running on a node, the maximum number of extra streaming requests it may handle is also different. For a fixed number of transcoding processes (assuming the number is N), we determine the maximum number of streaming requests Y_s in the following steps. First, we duplicate and concatenate short videos into long videos. Then we transcode N of those concatenated videos in parallel, while at the same time do streaming on that node and find the maximum number of requests it may handle. We ensure that no transcoding processes terminate before all the streaming have finished. This allows us to calculate number of hours of videos transcoded per second (denoted as Y_t), in addition to streaming Y_s videos. Here we use the maximum possible Y_s along each Y_t and N, since it is hard to fairly define the optimal combination of them. We measure the average power draw P_u, and calculate per second cost C_u with

$$C_u = P_u C_e + \frac{C_{hd}}{T}. \tag{4}$$

This results in a series of tuple (N, Y_s, Y_t, P_u, C_u). It means that when there are N parallel transcoding processes, the node can handle Y_s extra video streaming requests, while at the same time transcode Y_t hours of videos per second. Under such load it draws P_u power and costs C_u per second.

It is unlikely that any Y_s/Y_t equals to X_s/X_t. We always multiplex as much workloads as possible and handle the remaining workload with extra node. This idea is represented by

$$\begin{aligned} P_{union} &= \begin{cases} p_s(Y_s - X_s\frac{X_t}{Y_t}) + P_u\frac{X_t}{Y_t}, & \text{if } \frac{X_s}{Y_s} \geq \frac{X_t}{Y_t} \\ P_u\frac{X_s}{Y_s} + p_t(Y_t - X_t\frac{X_s}{Y_s}), & \text{otherwise} \end{cases} \\ C_{union} &= \begin{cases} c_s(Y_s - X_s\frac{X_t}{Y_t}) + C_u\frac{X_t}{Y_t}, & \text{if } \frac{X_s}{Y_s} \geq \frac{X_t}{Y_t} \\ C_u\frac{X_s}{Y_s} + c_t(Y_t - X_t\frac{X_s}{Y_s}), & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

Here we make some further explanation to the power equation. The first case is $X_s/Y_s \geq X_t/Y_t$, which means if we use enough union nodes to cover the transcoding workload, although some streaming workload is finished at the same time, there are still extra streaming workload to handle. The number of union nodes is dominated by the transcoding workload and calculated as X_t/Y_t. The extra streaming workload is $Y_s - X_s X_t/Y_t$, which is handled by extra nodes to do streaming only. The opposite case is similar and the cost is calculated in the same way.

Finally, Table I summarizes meanings of all the symbols we use in above equations. Here C_e is 0.4883 RMB/kWh and 1 RMB = 0.1577 USD. T is three years, which is adopted in previous work like [11].

Table I: Meaning of symbols

Var	Description
w	(subscript) either s or t, corresponding to the streaming or transcoding module, respectively
m	(subscript) any of $homo$, $hetero$ or $union$, corresponding to the mapping mode
M_w	performance metric; for the streaming module, it's the number of concurrent streaming requests; for the transcoding module, it's the number of hours of videos transcoded per second
X_w	performance metrics required by the SLA
Y_w	performance metrics when multiplexing the two modules
p_w	single-noded power performance ratio
P_w	single-noded measured power
P_u	single-noded measured power when multiplexing
P_m	power draw on achieving some SLA, when in this mapping mode
c_w	single-noded cost performance ratio
C_u	single-noded cost when multiplexing
C_m	overall cost on achieving some SLA, when in this mapping mode
C_e	electricity price
C_{hd}	hardware acquisition cost
T	expected hardware lifetime

Table II: VOD Applications

OS	Streaming	Transcoding	Source
Win	Helix Server	Helix Producer	Commercial products
Linux	VLC	FFmpeg	Open source

IV. Experiment Setup

Our experiment uses two VOD applications. We measure the power and cost efficiency on combinations of three types of processors and three types of disks.

A. Software

Table II summarizes the two VOD applications used in our experiment. Helix Universal Media Server and Helix Producer are commercial products from RealNetworks, Inc. These proprietary softwares are in use at University of Edinburgh, Thomas Telford School, University of Hawaii, etc. We run Helix softwares on Windows Server 2008 R2. Our second workload comes from two instances of open source software: VLC media player and FFmpeg. Despite a well know multimedia player, latest VLC also has built-in on-demand video streaming support. They are deployed on Ubuntu Server 12.04.1 LTS. We use up to twenty PCs as clients in the streaming experiment. The media players on clients are VLC.

B. Hardware Platforms

Nodes in our experiments are classified into servers, PCs and netbooks, according to their computing power. Each node is equipped with a number of SSDs or HDDs. We pair servers with SAS drives and couple PCs and netbooks with SATA drives. Altogether the platform space contains six types of nodes.

Table III lists specifications of the hardware. Originally the netbook has a 100Mbps Ethernet card. We arm it with a USB 2.0 Gigabyte adapter which sustains 240 Mbps throughput under performance test using `netperf` under Linux.

C. SLA From YouTube

We generate an SLA from public statistics of YouTube [12]. At the time we write this, 4 billion hours of videos are watched per month on YouTube. That is 5,600,000 (=4,000,000,000/(30*24)) concurrent streaming requests on average. 72 hours of videos are uploaded per minute. Videos are invariably transcoded into a single format [13]. Most videos can be watched in four resolutions (240/360/480/720p). We assume a transcoding factor of 5. That is transcoding 6 (=5*72/60) hours of videos in every second. Therefore the target SLA is $(X_s = 5,600,000, X_t = 6)$. We believe X_s is accurate while X_t is just an approximation because different codecs may affect the amount of work of transcoding. Anyway, the ratio X_s/X_t approximates the real case in order of magnitude.

Table III: Specifications of hardware configurations.

Node Type	*Server*	*PC*	*Netbook*
Model	Dell PowerEdge R410	DIY	Asus EeePC 1015PW
CPU Model	Xeon E5620	Core 2 Q6600	Atom N570
Clk. Speed(GHz)	2.40	2.40	1.66
#Sockets	2	1	1
#Cores	8	4	2
#Threads	16	4	4
TDP(W)	160	105	8.5
Mem(GB)	4	2	2
NIC(Mbps)	1000	1000	240
Avail. Disks	SSD, SAS	SSD, SATA	SSD, SATA
#Slots	4	6	1

(a) processors

Disk Type	*SSD*	*SAS*	*SATA*
IOPS	$> 10^4$	210	≈ 100
Capacity(GB)	120	300	1000

(b) disks

V. Experiment Results

This section illustrates experiment results.

A. Streaming Workload

Fig. 2 depicts the performance and power of the streaming module in Windows. SSDs invariably outperform traditional

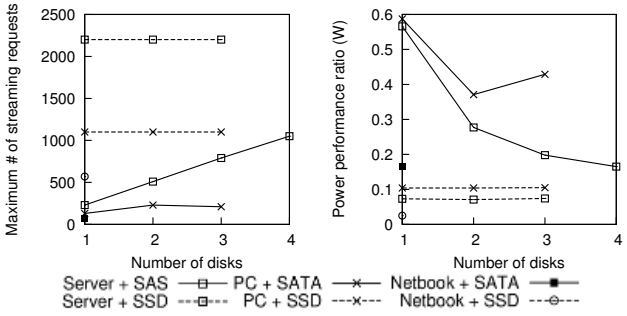


Figure 2: Performance and power performance ratios. (Helix Server in Windows)

disks no matter which processor they are coupled with. More powerful processor gains greater performance, while wimpy nodes with SSDs are most power efficient. Only one SSD is enough to shift the bottleneck away from I/O components. Fig. 2 shows that a wimpy node with an SSD can serve more than 500 streaming requests in Windows. In Linux, the number is 350[1]. We observed number of context switches sustained 40 – 60K per second in Linux. Such large number indicates implementation defects of the streaming

[1]Benchmarking details available at http://pacman.cs.tsinghua.edu.cn/~fredfsh/hpcc2013/benchmark.tgz.

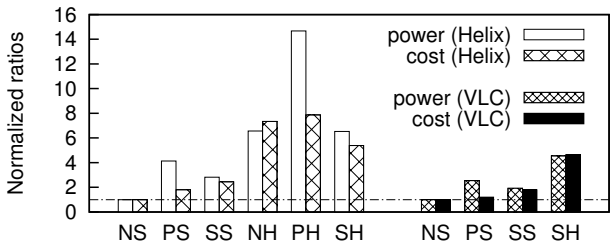


Figure 3: Power performance ratio and cost performance ratio of different platforms running the streaming module. The first letter denotes node type: N(etbook), P(C), S(erver), while the second letter denotes disk type: S(SD), H(DD).

application or Linux itself, which leads to the gap between the two operating systems. Adding more SAS disks to the server results in higher power efficiency, while two SATA drives are the best choice for a PC. Fig. 3 compares the power and cost efficiency of different platforms running the streaming module. In Linux, we didn't deploy the streaming module on PC/netbook with SATA drives because these are likely to be the poorest configurations. Fig. 3 confirms that wimpy nodes with SSDs are the most power and cost efficient platform to deploy the streaming module.

B. Transcoding Workload

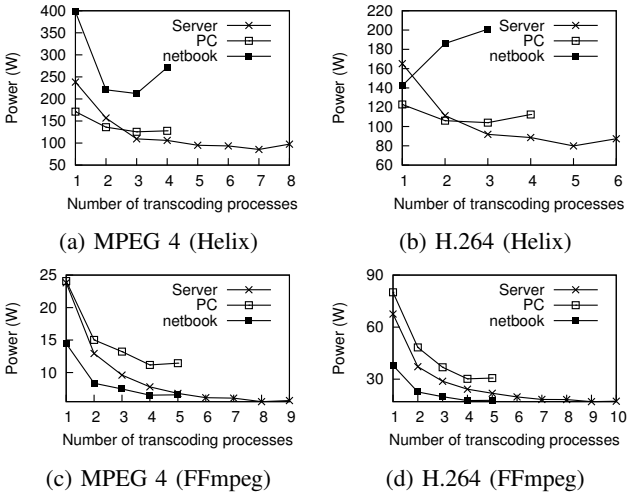
Fig. 4 depicts the power performance ratio of running the transcoding module. Disk type has negligible effect on power and cost performance ratio so we omit it here for brevity. The data illustrated here are SSD-based. In Windows, more powerful processor achieves higher power efficiency. In Linux, wimpy nodes have comparable power efficiency as servers, both better than PCs. Using different codes shows consistent results. Fig. 5 compares power and cost efficiency of different platforms running the transcoding module. Although PCs consume more power per unit work done, they are more cost effective than servers, except for using MPEG 4 in Linux.

C. Multiplexing Modules

Fig. 6 illustrates the power and cost to achieve the target SLA, by multiplexing the two modules on a server with SSDs. In Windows, the optimal choice is to multiplex three transcoding processes while streaming. In Linux, it is best to transcode six videos in parallel, in addition to streaming.

D. Comparison on Mapping Modes

Fig. 7 shows the power and cost of a cluster to achieve the target SLA, in different mapping modes. As SSDs are invariably more efficient than traditional harddrives, data related to homo mappings are SSD-based. Deploying the streaming module to wimpy nodes with SSDs and the transcoding module to servers reduces 28%/34% in power/cost per unit



(a) MPEG 4 (Helix) (b) H.264 (Helix)

(c) MPEG 4 (FFmpeg) (d) H.264 (FFmpeg)

Figure 4: The power draw of transcoding one hour of video per second.

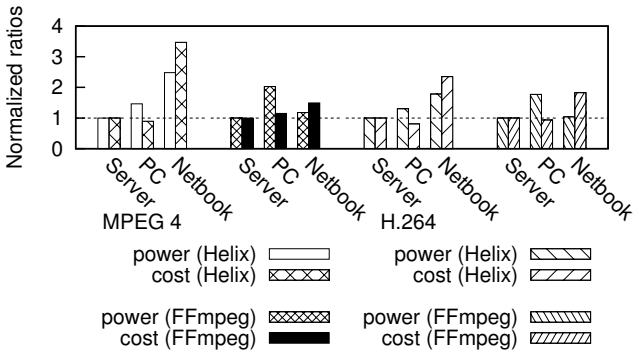


Figure 5: Power performance ratio and cost performance ratio of different platforms running the transcoding module.

work done, in Windows, compared with using merely Xeon servers. In Linux the number is 32%/33%. Despite this fact, such heterogeneous configuration is not so competitive among our chosen configurations. In Windows, multiplexing the two modules on servers with SSDs has comparable

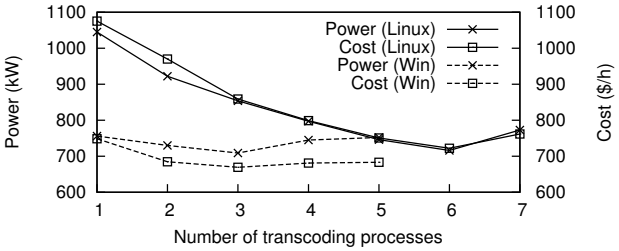


Figure 6: Power and cost of a cluster to achieve the target SLA, in union mode.

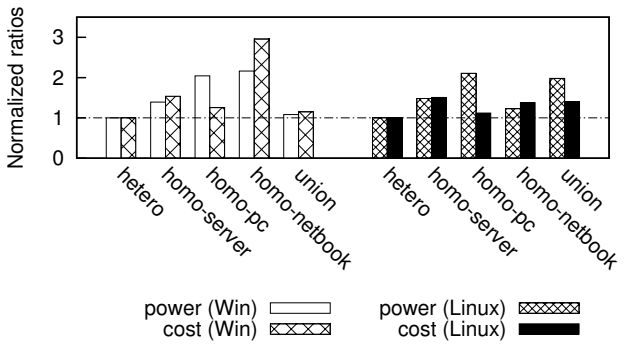


Figure 7: Power and cost of a cluster to achieve the target SLA.

cost and power effectiveness (less than 10% difference). However this union mode draws twice as much power as the heterogeneous solution and costs 40% more dollar in Linux, per unit work done. In Linux, using wimpy nodes only is the second best choice. But among the five configurations in Windows it is the most inefficient one.

VI. Related Work

Wimpy nodes with SSDs. The idea of coupling wimpy nodes and SSDs is raised by many researchers. Andersen et al. [1] design a key-value store on embedded CPUs with flash storage. Stuedi et al. [6] build a modified Memcached on nodes with 10 Gigabyte Ethernet adapters and processors of low clock frequency. Berezecki et al. [2] port Memcached to 64-core Tilera TILEPro64 platform. Szalay et al. [14] propose so-called *Amdahl blades* which combine wimpy nodes and SSDs to offer significantly high throughput and low energy consumption. On the other hand, Lang et al. [3] claim database applications don't perform well on wimpy clusters. Same authors propose an empirical method on cluster construction for database applications in [4]. Authors of FAWN [5] conduct experiments to explore which kinds of workloads are handled well (or not well) on FAWN-like platforms. To our best knowledge, we are the first to deploy VOD systems on various hardware configurations, including the FAWN-like platform. We also quantify the effectiveness of multiplexing different modules to same nodes.

Energy management in heterogeneous data center. There is much work on energy management considering the heterogeneity of a data center. Nathuji et al. [15] leverage power-management support from nodes themselves for power budgeting. Heath et al. [16] model the effect of running each workload on every type of node with certain power and performance metrics. Rusu et al. [17] present a cluster-wide QoS-aware technique that dynamically reconfigures the cluster to reduce energy consumption. CASH'EM [18] maximizes profit from data center's point of view. Bertini et al. [19] solve power-efficiency problem in heterogeneous web server clusters with the help of control theory. Mukherjee et al. [20] extend energy-management to the temporal dimension for virtualized heterogeneous data centers. Our experiment shows that energy reduction involves corporation of applications, operating systems, different components of hardware and application-to-node mapping modes. Any of these factors may affect power and cost efficiency.

VII. Conclusion

In this paper we perform a case study on finding the optimized configuration for VOD systems. We evaluate two VOD applications, in Windows and Linux respectively, on combinations of three types of processors and three types of disks, in three module mapping modes. The experiment result shows that, scheduling the streaming module to wimpy nodes with SSDs and the transcoding module to Xeon servers reduces no more than 35% power or cost per unit work done, compared with the server-only solution, in both Windows and Linux. Multiplexing different modules on servers achieves comparable power and cost reduction, in Windows. In Linux it is even worse than the traditional non-multiplexing server-only configuration. In Linux, using wimpy nodes for both modules is the second optimal configuration among our choices, but it is the poorest in Windows. These anti-intuitive and non-consistent results over different operating systems imply the intrinsic complexity of the problem. Finding an optimized configuration for a VOD system needs empirical results to support quantitative instead of qualitative analysis.

Acknowledgment

The authors would like to thank the anonymous reviewers whose inputs greatly improved the paper. Our work gets support partly from NSFC project 61133006, National High Technology Project 2012AA01A302, and National High-Tech Research and Development Plan (863 project) 2013AA01A215.

References

[1] D. Andersen, J. Franklin, M. Kaminsky, A. Phanishayee, L. Tan, and V. Vasudevan, "Fawn: A fast array of wimpy nodes," in *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*. ACM, 2009, pp. 1–14.

[2] M. Berezecki, E. Frachtenberg, M. Paleczny, and K. Steele, "Many-core key-value store," in *Green Computing Conference and Workshops (IGCC), 2011 International*. IEEE, 2011, pp. 1–8.

[3] W. Lang, J. Patel, and S. Shankar, "Wimpy node clusters: What about non-wimpy workloads?" in *Proceedings of the Sixth International Workshop on Data Management on New Hardware*. ACM, 2010, pp. 47–55.

[4] W. Lang, S. Harizopoulos, J. Patel, M. Shah, and D. Tsirogiannis, "Towards energy-efficient database cluster design," *Proceedings of the VLDB Endowment*, vol. 5, no. 11, pp. 1684–1695, 2012.

[5] V. Vasudevan, D. Andersen, M. Kaminsky, L. Tan, J. Franklin, and I. Moraru, "Energy-efficient cluster computing with fawn: workloads and implications," in *Proceedings of the 1st International Conference on Energy-Efficient Computing and Networking*. ACM, 2010, pp. 195–204.

[6] P. Stuedi, A. Trivedi, and B. Metzler, "Wimpy nodes with 10gbe: leveraging one-sided operations in soft-rdma to boost memcached," in *Proceedings of the 2012 USENIX conference on Annual Technical Conference*. USENIX Association, 2012, pp. 31–31.

[7] "Intel atom processor n570 (1m cache, 1.66ghz) specifications," http://ark.intel.com/products/55637/Intel-Atom-Processor-N570-(1M-Cache-1_66-GHz).

[8] "Intel xeon processor e5620 (12m cache, 2.40ghz, 5.86 gt/s intel qpi) specifications," http://ark.intel.com/products/47925/Intel-Xeon-Processor-E5620-(12M-Cache-2_40-GHz-5_86-GTs-Intel-QPI).

[9] A. Ku, "Endurance testing: Write amplification and estimated lifespan," http://www.tomshardware.com/reviews/ssd-520-sandforce-review-benchmark,3124-11.html.

[10] "Yokogawa – products – wt210/wt230 digital power meters," http://tmi.yokogawa.com/products/digital-power-analyzers/digital-power-analyzers/wt210wt230-digital-power-meters/.

[11] A. Greenberg, J. Hamilton, D. Maltz, and P. Patel, "The cost of a cloud: research problems in data center networks," *ACM SIGCOMM Computer Communication Review*, vol. 39, no. 1, pp. 68–73, 2008.

[12] "Youtube statistics," http://www.youtube.com/t/press_statistics.

[13] "Youtube video formats," http://www.mediacollege.com/video/internet/youtube/formats.html.

[14] A. Szalay, G. Bell, H. Huang, A. Terzis, and A. White, "Low-power amdahl-balanced blades for data intensive computing," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 1, pp. 71–75, 2010.

[15] R. Nathuji, C. Isci, and E. Gorbatov, "Exploiting platform heterogeneity for power efficient data centers," in *Autonomic Computing, 2007. ICAC'07. Fourth International Conference on*. IEEE, 2007, pp. 5–5.

[16] T. Heath, B. Diniz, E. Carrera, W. Meira Jr, and R. Bianchini, "Energy conservation in heterogeneous server clusters," in *Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming*. ACM, 2005, pp. 186–195.

[17] C. Rusu, A. Ferreira, C. Scordino, and A. Watson, "Energy-efficient real-time heterogeneous server clusters," in *Real-Time and Embedded Technology and Applications Symposium, 2006. Proceedings of the 12th IEEE*. IEEE, 2006, pp. 418–428.

[18] J. Burge, P. Ranganathan, and J. Wiener, "Cost-aware scheduling for heterogeneous enterprise machines (cashem)," in *Cluster Computing, 2007 IEEE International Conference on*. IEEE, 2007, pp. 481–487.

[19] L. Bertini, J. Leite, and D. Mossé, "Power optimization for dynamic configuration in heterogeneous web server clusters," *Journal of Systems and Software*, vol. 83, no. 4, pp. 585–598, 2010.

[20] T. Mukherjee, A. Banerjee, G. Varsamopoulos, S. Gupta, and S. Rungta, "Spatio-temporal thermal-aware job scheduling to minimize energy consumption in virtualized heterogeneous data centers," *Computer Networks*, vol. 53, no. 17, pp. 2888–2904, 2009.