Rethinking High Performance Computing System Architecture for Scientific Big Data Applications

Yong Chen*, Chao Chen*, Yanlong Yin†, Xian-He Sun†, Rajeev Thakur‡, William D Gropp§

*Department of Computer Science, Texas Tech University, Email: yong.chen@ttu.edu, chao.chen@ttu.edu
†Department of Computer Science, Illinois Institute of Technology, Email: yyin2@ttu.edu, sun@ttu.edu
‡Mathematics and Computer Science Division, Argonne National Laboratory, Email: thakur@mcs.anl.gov
§Department of Computer Science, University of Illinois Urbana-Champaign, Email: wgropp@illinois.edu

***Abstract*—The increasingly important data-intensive scientific discovery presents a critical question to the high performance computing (HPC) community - how to efficiently support these growing scientific big data applications with HPC systems that are traditionally designed for big compute applications? The conventional HPC systems are computing-centric and designed for computation-intensive applications. Scientific big data applications have growlingly different characteristics compared to big compute applications. These scientific applications, however, will still largely rely on HPC systems to be solved. In this research, we try to answer this question with a rethinking of HPC system architecture. We study and analyze the potential of a new decoupled HPC system architecture for data-intensive scientific applications. The fundamental idea is to decouple conventional compute nodes and dynamically provision as data processing nodes that focus on data processing capability. We present studies and analyses for such decoupled HPC system architecture. The current results have shown its promising potential. Its data-centric architecture can have an impact in designing and developing future HPC systems for growingly important data-intensive scientific discovery and innovation.**

I. Introduction

Many scientific simulations in critical areas, such as climate sciences, astrophysics, computational chemistry, computational biology, and high-energy physics, are becoming increasingly data intensive [1, 2]. These applications manipulate a large amount of data relative to the amount of computation they perform, and often transfer large amounts of data to and from storage systems. Some application teams have already begun to process terabytes or tens of terabytes of data in a single simulation run. For example, 12 out of 25 INCITE applications run on the Department of Energy leadership computing system at Argonne National Laboratory several years ago have already processed datasets in the terabyte range [3, 4].

Meanwhile, the data collected from instruments for scientific discoveries and innovations are increasing rapidly too. For example, the Global Cloud Resolving Model (GCRM) project, part of Department of Energy's Scientific Discovery through Advanced Computing (SciDAC) program [5], is built on a geodesic grid that consists of more than 100 million hexagonal columns with 128 levels per column. These 128 levels will cover a layer of 50 kilometers of atmosphere upwards from the surface of the earth. For each of these grid cells, scientists need to store and predict data like the wind velocity, temperature, pressure, etc. Most of these global atmospheric models currently process data in a 100-kilometer scale (the distance on the ground); however, scientists desire higher resolution and finer granularity, which can lead to even larger sizes of datasets. In addition, the proliferation of sensing technologies and the increased usage of remote sensors are also generating huge amount of data than ever before.

Both experimental data and simulation data are rapidly increasing, and these scientific discoveries and innovations have exhibited a critical data-intensive computing need, creating the "big data" computing era in recent years [6–8]. High Performance Computing (HPC) is a strategic tool during the process of scientific discoveries and innovations. These increasingly data-intensive scientific problems will still rely on HPC systems to compute, analyze, and answer the problems, which is an essential process of understanding the phenomenon behind the data. The conventional HPC systems, however, are *computing-centric* and designed for *computation-intensive applications*. They are not ready and can have inherent limitations when used for solving increasingly data-intensive problems. How to design and develop HPC system architectures for efficient processing ever-growing scientific data has become a key challenge in the big data computing era.

In this research, we revisit the HPC system architecture and study the impact of a new *decoupled high performance computing system architecture* for data-intensive sciences. Such a study can shed light on designing and developing next-generation HPC systems for growingly critical data-intensive computing. A decoupled system architecture has the notion of the separation of compute nodes and data processing nodes. The novelty of the decoupled HPC system architecture is that, instead of dedicating the dominant investment to compute nodes as in the conventional HPC system architecture, the new investment is decoupled into compute nodes and data nodes. These data nodes are designed to handle data-intensive operations with a mission of minimizing data movement. Compute nodes, as in the conventional architecture, handle computation-intensive operations. Scientific big data applications can be mapped to such a decoupled HPC system architecture and are executed in a decoupled but fundamentally more efficient manner with the collective support from data nodes and compute nodes. Ideally these data nodes and compute nodes can be dynamically configured and provisioned depending on application-specific characteristics, e.g. the intensity of data accesses. This research focuses on modeling and analysis of a decoupled HPC system architecture. A fundamental question we try to answer in this research is that how HPC system architecture should be designed and developed to best support data-intensive scientific computing. Our idea of a decoupled system architecture is a new thinking of HPC system architecture design and development when

data access is as important as computation. A decoupled HPC system architecture can be such a possible solution. The current results have shown that it is promising and has a potential.

The rest of this paper is organized as follows. Section II presents the idea and framework of a decoupled HPC system. Section III introduces the modeling and analysis of the decoupled HPC system architecture. Section IV reviews existing studies in related areas and compares with our work. Section V concludes this study and discusses future work.

II. Decoupled High Performance Computing System Architecture: Motivation and Overview

In scientific applications, data is commonly represented with a multi-dimensional array-based data model. For instance, the widely used Community Earth System Model (CESM) software package consists of four separate modules simultaneously simulating the earths atmosphere, ocean, land surface and sea-ice, and each module uses the multi-dimensional arrays data model [9]. A common example is a 3-dimensional temperature data with longitude, latitude, and time dimensions. It is often needed to compute the moving average, median, lowest and highest temperature with specified conditions such as areas and periods of time. Such computed results will be further correlated with the computed results from other parameters, such as the humidity and wind velocity, to predict weather conditions.

The current way of conducting such processing is to read the required data (e.g., a sub-array of interested area) from storage servers to compute nodes, perform computations on desired data with specified conditions, such as those data shown in shaded area, and then write the output back to storage. For CESM, an experimental test shows that the data access and movement time for the calculation of the moving average, median, lowest and highest degrees can occupy 88.2%, 95.4%, 96.6%, and 96.6% of the total execution time on a cluster, where 128GB of data are retrieved to 272 nodes for processing.

CESM clearly has data retrieval and processing phases and computing and simulation phases, as many scientific big data applications do. The basic idea of the new decoupled HPC system architecture is to change the conventional architecture to handle these two phases differently on different nodes. Such an architecture decouples nodes into compute nodes and data processing nodes. These nodes are mapped with computation-intensive operations and data-intensive operations respectively. Computation-intensive operations are executed on massive compute nodes. Data-intensive operations are executed on dedicated data processing nodes. In other words, the decoupled architecture reshapes the current pattern of retrieve - compute - store cycles into retrieve (generate) - reduce - compute - reduce - store cycles as shown in Figure 1, where the reduce phases are designed to conduct offloaded data-intensive operations and reduce data size before moving data across the network. These retrieval, reduce, compute, and store phases can be pipelined to overlap the I/O, communication, and computation times. From one point of view, the decoupled architecture is an enhanced framework of MapReduce [10], where the reduce is not conducted by one node with its local storage, but a set of (data) nodes and the global storage, so that parallel computing features can be maintained. From another point of view, the data nodes are the data-access accelerators, to speed up data accesses and reduce data size before sending data across the network.

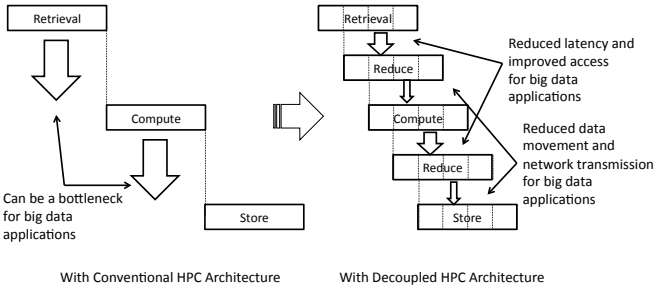


Fig. 1. Comparison of HPC Architectures

The decoupled HPC system architecture is shown in Figure 2. This architecture decouples the nodes of conventional investment into data nodes and compute nodes. Data nodes are further decoupled into compute-side data nodes and storage-side data notes. Compute-side data nodes are compute nodes that are dedicated for data processing. Storage-side data nodes are specially designed nodes that are connected to file servers with fast network. Compute-side data nodes reduce the size of computing generated data before sending it to storage nodes. Storage-side data nodes reduce the size of data retrieved from storage before sending it to compute-side data nodes. Writes will go through compute-side data nodes, whereas reads will go through the storage-side data nodes. Data nodes can provide simple data forwarding without any data size reduction, but the idea behind data nodes is to let data nodes conduct the offloaded data-intensive operations and optimizations to reduce the data size and data movement.

In this research, we focus on studying the implications of such a decoupled HPC system with an assumption that operations from applications can be decoupled into computation-intensive and data-intensive operations respectively. Our study focuses on how to design and configure the decoupled HPC system architecture considering scientific big data applications features, such as the intensity of data accesses and characteristics of computing and data accesses. Performance tools can provide information and guidance to understand application computing and data-access characteristics and intensity of data accesses. For instance, we have developed an IOSIG performance tool to find the I/O access signature (patterns) of an application [11, 12]. We have extended IOSIG to IOSIG+ to identify the data-intensive phases and computation-intensive phases. I/O dependency analysis can also be used to separate the phases [13], and can be used to find the hot data which lead to operations that should be conducted on the data nodes.

The decoupled HPC system architecture changes the current architecture by balancing the computation and data-access capabilities for data-intensive applications. This new architecture separates computation-intensive operations and data-intensive operations and handles them concurrently and in a coordinated manner, but on different hardware and software environments for best performance. The architecture configuration is flexible. At one extreme, it could have no data nodes. In that case, it is the traditional HPC architecture. At another extreme, the compute nodes could be simple SIMD processing elements. In that case, the compute nodes are more

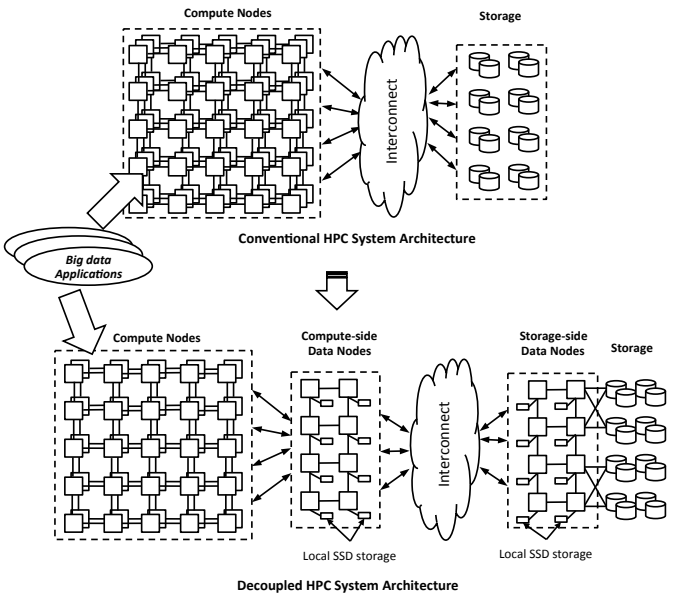


Fig. 2. Decoupled High Performance Computing System Architecture

like computing accelerators. In this research, while we will study different configurations, the focus is on understanding such decoupled system architecture and impact on scientific big data applications. We will discuss details in the following section from modeling and analyses.

III. Decoupled High Performance Computing System: Modeling and Analysis

In this section, we present detailed modeling and analyses of the decoupled HPC system. We first introduce the performance model used to analyze the potential of system architecture designs, and then present detailed studies from both architecture configurations and also application features.

A. Modeling and Comparison of HPC Systems

There is no argument that many performance models for parallel computing and HPC systems exist, such as the well-known Amdahl's model. A common limitation of these models, however, is that they primarily focus on computation part of applications to direct building HPC systems for computation-intensive applications. Our analyses are based on a simple but effective model that captures both computation and data access, which is especially needed for analyzing HPC system architectures for data-intensive applications given the growing importance.

Different from previous models, our model takes data workload into consideration to analyze the system performance. This model assumes that the execution of an application logically follows the model illustrated in the Figure 3, which is a simplified abstraction and generally holds for many applications without optimizations like pipelining, asynchronous I/O, etc. This performance model is based on an assumption that applications conduct interleaved and periodic computations and data accesses. In each phase, the total workload W contains two parts: computation workload part (W_C) and data workload part (W_D). For simplicity, we assume that W_C and W_D are the same for each phase. The model and analysis can be applicable to a general case that W_C and W_D are different for each phase, which means that there exist $W_C i$ and $W_D i$ for each phase. Thus, the entire workload of an application can be derived and expressed as:

$$W = (W_C + W_D) \cdot m \tag{1}$$

where m is the number of phases.

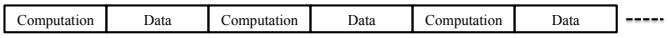


Fig. 3. Execution Model of An Application on High Performance Computing Systems

To derive the detailed analysis of the new decoupled HPC system architecture, the model also introduces several notations to characterize dominant applications and systems parameters, as shown in Table I.

TABLE I
Parameter Notations

n	the number of compute nodes
r	the ratio of data nodes compared to compute nodes
b	the network bandwidth of each compute node in the conventional and decoupled architectures
b_h	the network bandwidth of each data node in the decoupled system architecture
λ	the ratio between the network bandwidth of compute nodes and data nodes, i.e. b_h/b
$f_{op}(x)$	the result workload of an offloaded data-intensive operation *op* with input x raw data workload, or the computation workload (termed as *seed workload*) for generating x size workload
η	the ratio of data-intensive operation's computation workload compared to the whole computation workload of an application
γ	the ratio of the result workload or seed workload of an offloaded data-intensive operation compared to the raw data workload
α	the ratio between the computation workload and data workload

In conventional HPC system architecture, the total workload is divided and executed on n compute nodes. Applications process data on compute nodes and conduct I/O operations directly with storage nodes. Thus, the execution time (T) and the performance (P) of a conventional HPC system can be modeled as:

$$T = (\frac{W_C}{n} + \frac{W_D}{n \cdot b}) \cdot m \tag{2}$$

$$P = \frac{1}{T} \tag{3}$$

With the decoupled HPC system architecture, data nodes (including compute-side data nodes and storage-side data nodes) are specially designed for data-intensive operations. Data nodes are connected to compute nodes (for compute-side data nodes) or storage nodes (for storage-side data nodes) through a high-speed network. Data-intensive operations can be decoupled and offloaded to data nodes according to their features. In a read-intensive situation, where the application retrieves a large volume of data and then performs computations (such as SUM and k-means) to obtain a small-size result for further processing, this operation can be offloaded to storage-side data nodes and only the result is returned to compute nodes. The decoupled system architecture reduces considerable data movement and improves the usage of the precious network bandwidth. In a write-intensive situation, the computations that generate a large volume of data (we term as "seed operations") can be offloaded to storage-side data nodes.

These computations are performed on storage-side nodes to generate data in place instead of generating data on compute nodes and moving through the network. The execution time and the performance of the decoupled system architecture can thus be derived and defined as

$$T' = (\frac{W_C - W_{op}}{n\cdot(1-r)} + \frac{f_{op}(W_D)}{n\cdot(1-r)\cdot b} + \frac{W_D}{n\cdot r\cdot b_h} + \frac{W_{op}}{n\cdot r}) \cdot m \quad (4)$$

$$P' = \frac{1}{T'} \quad (5)$$

where W_{op} is the related computing workload of the data-intensive operation. It is directly related to data workload and can not be ignored. In the model, it can be expressed as $W_{op} = \eta \cdot W_C$. In practice, parameters, η, γ, λ, meet the following conditions:

$$0 < \eta < 1 \quad (6)$$

$$0 < \gamma < 1 \quad (7)$$

$$\lambda > 1 \quad (8)$$

To compare the performance of the decoupled HPC system architecture and conventional architecture, we can calculate the performance difference, defined as:

$$\Delta = P' - P = \frac{T - T'}{T \cdot T'} \quad (9)$$

If $\Delta > 0$, it means that the decoupled system architecture has better performance than the conventional architecture. Otherwise, the conventional architecture is better. Due to $T \cdot T' > 0$, the sign of the Δ will depend on $\rho = T - T'$. Based on the above assumptions and analyses, the equation for calculating ρ is derived as:

$$\rho = \frac{m}{n\cdot(1-r)\cdot r} \cdot [(2\cdot r\cdot\eta - \eta - r^2)\cdot W_C + \frac{\lambda\cdot r\cdot(1-r) - \gamma\cdot r\cdot\lambda - 1 + r}{\lambda\cdot b} \cdot W_D] \quad (10)$$

Another parameter, α, is introduced to represent the relationship between W_C and W_D and is defined as:

$$W_C = \alpha \cdot W_D \quad (11)$$

Equation (11) can be used to quantify the data-access intensiveness of an application. If $\alpha > 1$, which means the computation workload is larger than the data workload, the application can be considered as computation-intensive. Otherwise, if $0 < \alpha < 1$, the application can be considered data-intensive. With the parameter α, ρ can be derived as:

$$\rho = \frac{m\cdot W_D}{n\cdot(1-r)\cdot r} \cdot [(2\cdot r\cdot\eta - \eta - r^2)\cdot\alpha + \frac{\lambda\cdot r\cdot(1-r) - \gamma\cdot r\cdot\lambda - 1 + r}{\lambda\cdot b}] \quad (12)$$

In equation (12), all parameters, including both system-related and application-related parameters, are involved. This observation confirms the expectation that different applications need different system configurations to achieve the best performance. In general, given a system and an application, m, n, W_D are constant values, and they can not affect the system configurations. The rest of parameters will impact the system configurations. Thus, ρ' is introduced as follows and the rest analysis focuses on understanding the impact of these parameters of both system and applications' characteristics on the performance:

$$\rho' = \frac{1}{(1-r)\cdot r} \cdot [(2\cdot r\cdot\eta - \eta - r^2)\cdot\alpha + \frac{\lambda\cdot r\cdot(1-r) - \gamma\cdot r\cdot\lambda - 1 + r}{\lambda}] \quad (13)$$

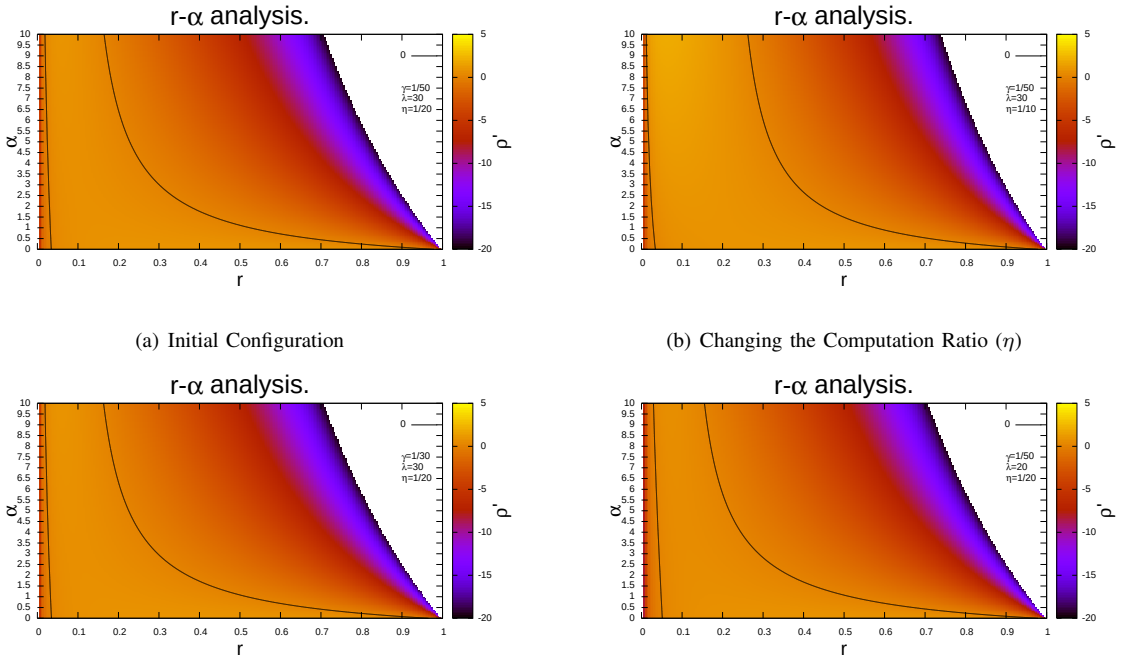
B. Analysis of the Decoupled HPC System Architecture

In this subsection, we focus on using the performance model discussed in the previous subsection to evaluate whether the new decoupled system architecture is more effective than the conventional architecture for scientific big data applications. This evaluation specifically focuses on two parameters, r and α, because they represent the system architecture and applications characteristics respectively. First, r denotes the ratio of the data nodes in the decoupled system architecture. Different values of r represent different system configurations. For example, if $r = 0$, there will be no data nodes in the system, and the architecture will be completely the same with the conventional architecture. Second, α in the model is used to measure the intensity of data accesses in an application. If $0 < \alpha < 1$, it implies more data workload than computation workload; thus, the application is more data-intensive. Otherwise, if $\alpha > 1$, it implies that the application is more computation-intensive. Therefore, α is the parameter that characterizes applications features.

The goal of this evaluation is to compare the decoupled HPC system architecture and the conventional architecture. Variable ρ' is used to evaluate which one is better. If $\rho' < 0$ holds under any configuration for r and α, it means that the conventional architecture outperforms the decoupled architecture for both data-intensive and computation-intensive applications. Otherwise, it confirms that the decoupled architecture achieves better performance than the conventional architecture under evaluated configurations of systems and applications.

Figure 4 reports the impact of r and α on performance. It plots four $r - \alpha$ graphs given various settings, where r varies from 0 to 1, and α varies from 0 to 10. To better compare the decoupled system architecture and the conventional architecture, these four graphs were plotted with different configurations for λ, η, and γ, as indicated on each graph. Different colors are used to represent different values of ρ'. To assist comparison, we specifically draw a contour line 0, which represents $\rho' = 0$, to help identify these areas that represent $\rho' > 0$ or $\rho' < 0$. The contour line 0 splits each graph into three areas: the left-most and right-most areas that represent $\rho' < 0$, and middle area that represents $\rho' > 0$. The left-most area is caused by the model derivation, because when $r \to 0$, $\frac{W_D}{n\cdot r\cdot b_h}$ in T' will be sufficiently close to ∞, which is impossible to happen in practice.

Comparing Figure 4-a and Figure 4-b, we can find that when the value of η increases, the space of middle area increases too, which means the decoupled system architecture is beneficial for more cases. This is because when η increases, the computation workload conducted on compute nodes will be decreased. Figure 4-c plots the results when increasing value γ. Parameter γ represents the offloading efficiency of a data-intensive operation in reducing the data movement. The lower the the value of γ is, the more data reduction and the less time in transferring data. Compared to Figure 4-a, it shows that γ has little impact on the system configuration. It is not necessary to reconfigure the ratio of data nodes when



(a) Initial Configuration

(b) Changing the Computation Ratio (η)

(c) Changing the Ratio of Result Workload (γ)

(d) Changing the Ratio of Network Bandwidth (λ)

Fig. 4. Analysis of the Decoupled High Performance Computing System Architecture[1]

γ is changed. Similarly, Figure 4-d plots the results when λ is changed, which has minor impact on system configuration as well.

In summary, despite the model derivation, the results shown in Figure 4 illustrate that the decoupled HPC system architecture is better than the conventional architecture for data-intensive applications in most cases (where $\alpha < 1$). Besides, the results also show that when $\alpha > 1$, the decoupled architecture can still be configured with suitable ratio of data nodes to achieve better performance than the conventional architecture. In addition, these graphs imply that the ratio of data nodes should be configured according to applications' characteristics. When applications tend to be more computation-intensive, the number of data nodes should be reduced accordingly. As observable from the graphs, when the α increases from 0 to 1, the value of r is decreased from 1 to around 0.55. When the α increases from 1 to 10, the value of r is decreased slowly, from 0.55 to 0.23. A dynamically configurable architecture would be an ideal solution to best support applications with considering applications' characteristics.

C. Analysis from Systems' Perspective

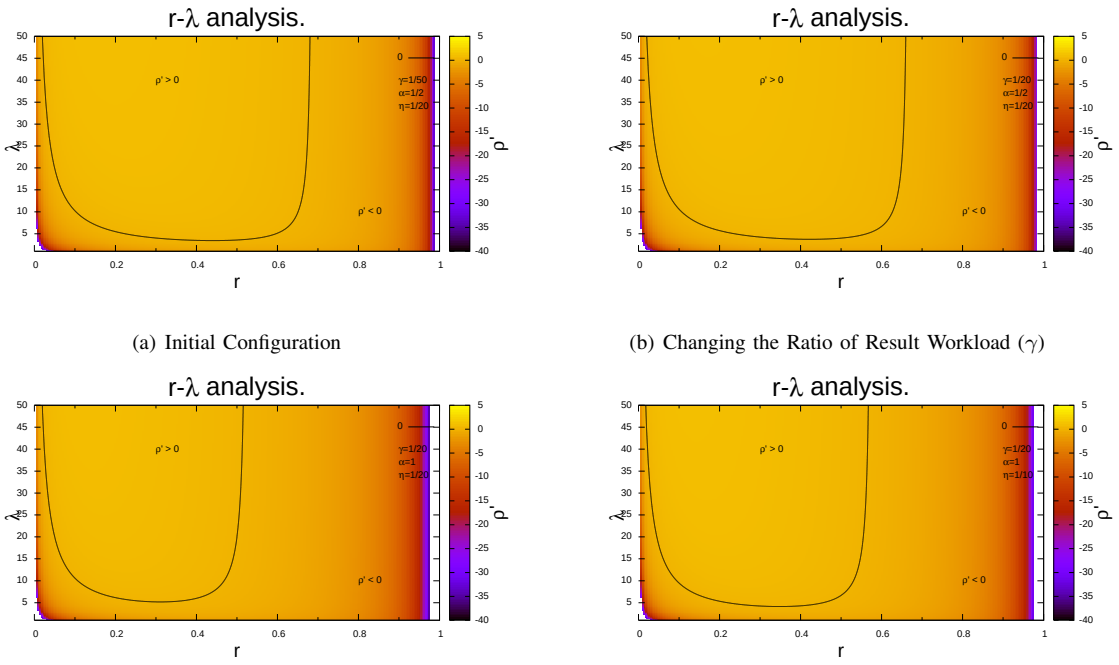
As analyzed in subsection III-B, a dynamically configurable system architecture is a preferred solution. In practice, such an ideal solution is hard to be achieved. One critical challenge is that, modern high performance computing systems are designed to run many scientific applications simultaneously. A dynamic configuration for a specific application would be challenging for other applications. Therefore, how do we design and develop a fixed system configuration for various data-intensive applications? In this subsection, we try to answer this question. We will analyze and show how to configure a high performance computing system with the decoupled architecture for scientific big data applications without prior knowledge of applications.

To answer this question, we need to analyze the system-related parameters. In the performance model, r and λ are two system-related parameters. Parameter r represents the ratio of data nodes configuration. It illustrates how to deploy data nodes and compute nodes. Parameter λ represents the network configuration. It shows the network requirement of a decoupled system architecture. In practice, λ is completely determined by the interconnection deployed and physical configuration. If data nodes are deployed physically closer to storage node, the λ is expected to be higher.

Since we do not have the prior knowledge and need to support multiple applications, we use different applications to find an overlap area to determine the optimal values for r and λ. Figure 5 plots four $r - \lambda$ graphs with changing values of application related parameters, including γ, α, and η. We varied the values of these parameters one by one, and observe how these parameters can affect the system configuration. Various values of γ, α, and η represent different applications. Since we focus on data-intensive applications, we make α vary between 0 and 1.

In general, these figures show that, the decoupled HPC system architecture can achieve better performance for scientific big data applications when the values of λ and r range in an enclosed area. As can be observed from the graphs, when γ increased from $\frac{1}{50}$ to $\frac{1}{20}$, the configuration for r and λ only changes by a small amount, which is consistent with the analysis in subsection III-B. Besides, when the value of α increases from $\frac{1}{2}$ to 1, the preferred range for r shrinks from $(0.1, 0.63)$ to $(0.1, 0.5)$. This observation is reasonable, because when α increases, the application tends to be more computation-intensive and it needs more compute nodes to

[1]We plotted in color for the best result. Please read from the electronic file or color printouts. The rest of figures were plotted in a similar setting.



(a) Initial Configuration

(b) Changing the Ratio of Result Workload (γ)

(c) Changing the Ratio of Computation Workload (α)

(d) Changing the Ratio of Network Bandwidth (λ)

Fig. 5. Analysis from Systems' Perspective

conduct computations. In addition, if we keep increasing the value for η, the preferred range for r will expand slightly, because the computation workload conducted on compute nodes would be decreased as η increases.

In summary, we can conclude that, when designing and deploying a high performance computing system for data-intensive applications, there are two rules we can use from the study of this research:

1) Ensure $\lambda \cdot r > 1$
2) Choose a ratio of data nodes in the range of $(0.2, 0.4)$

D. Analysis from Applications' Perspective

In the performance model, three parameters are used to represent applications' characteristics: α, γ, and η. Parameter α is used to quantify the data-access intensity of an application and identify whether an application is data-intensive $(\alpha < 1)$ or computation-intensive $(\alpha > 1)$. Parameter γ is used to identify the offloading effectiveness of the data-intensive operation in reducing the data size. Parameter η represents the computation workload of data-intensive applications.

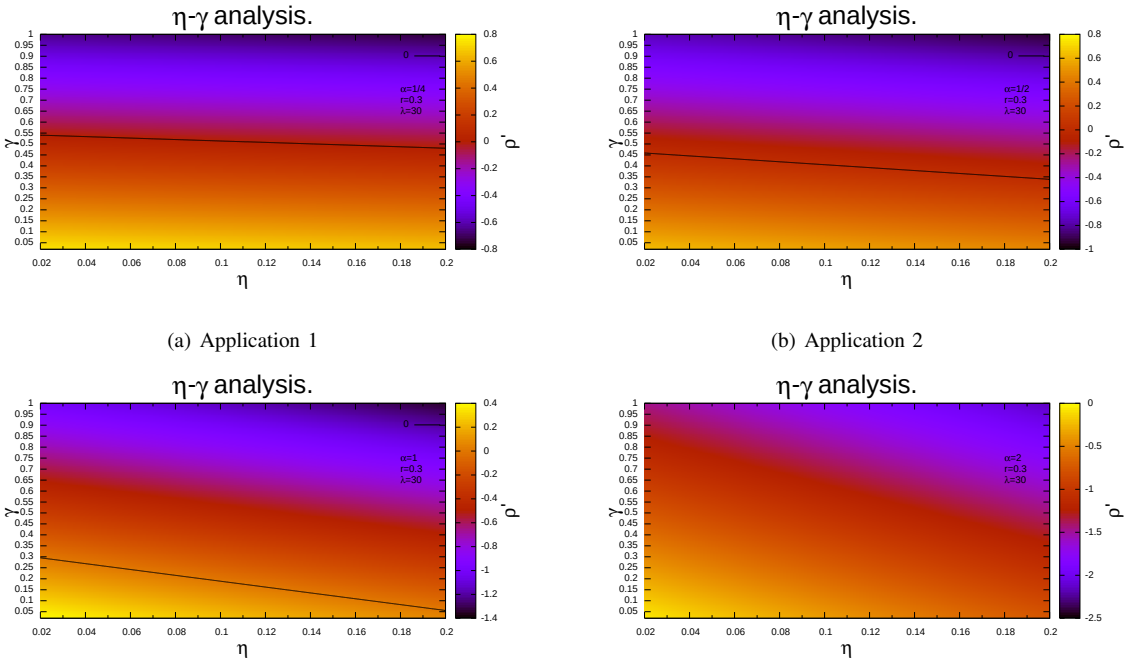
In subsection III-B, we have made several conclusions about the relationship between r and α. We conclude that the decoupled HPC system architecture is a better solution for data-intensive applications. In addition to α, two other parameters, γ and η, also represent applications' characteristics. Even though we find that both parameters have minor impact on system configurations through the earlier analysis, we present a more detailed analysis in this subsection. We attempt to answer the question: can the new decoupled HPC system architecture be deployed for all scientific big data applications?

To answer this question, we evaluated four different applications with different intensity of data accesses (indicated by the value of α). A fixed system configuration is used for this evaluation, and the values for system-related parameters were chosen based on the analysis in the prior subsection. Figure 6 plots four $\eta - \gamma$ graphs to represent four different applications respectively. These figures show that, compared to η, parameter γ has a greater impact on the system configuration. Although changing slightly according to different data-access intensity, it generally requires $\gamma < 0.3$ for most data-intensive applications. This observation means that applications with data-intensive operations that can reduce the data size to lower than $\frac{3}{10}$ of the raw data size can especially benefit from the decoupled HPC system architecture. For computation-intensive applications (Application 4 in the figure), the configuration of the decoupled system architecture in this evaluation did not provide better performance than the conventional architecture. However, according to the analysis in subsection III-B, we can configure a lower value of r for computation-intensive applications to improve the performance with the decoupled system architecture, as shown in Figure 7. It restricts the values of η and γ though, especially for η. In fact, this observation is straightforward because when an application becomes more computation-intensive, the value of η will reduce automatically.

In summary, scientific big data applications that have low computation ratio ($\eta < 0.2$) and can reduce the data size effectively ($\gamma < 0.3$) can especially benefit from the decoupled HPC system architecture. Meanwhile, computation-intensive applications can also benefit from the decoupled system architecture with choosing a proper configuration ratio, which confirms the conclusion in subsection III-B.

IV. Related Work

Extensive studies have focused on improving the performance of HPC systems for data-intensive applications at various levels. At the hardware level, the emerging nonvolatile storage-class memory (SCM) devices such as flash-memory based solid-state drives (SSDs) and phase-change memory



(a) Application 1

(b) Application 2

(c) Application 3

(d) Application 4

Fig. 6. Analysis from Applications' Perspective

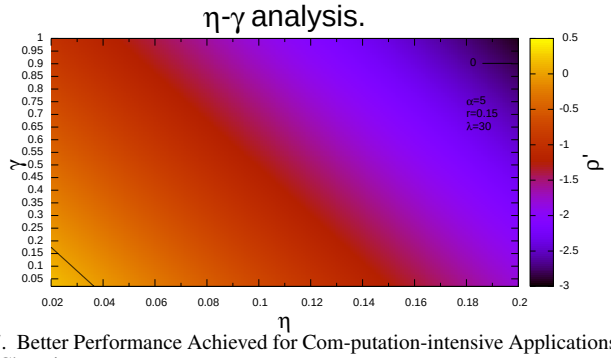


Fig. 7. Better Performance Achieved for Com-putation-intensive Applications with Changing r

(PCRAM) can provide more promising performance than hard disk drives (HDDs) [14, 15], especially for random accesses [14, 16]. However, they cannot reduce the data movement across the network. They help mitigate the performance gap between processors and I/O devices but will not be able to address the issue of large volume of data movement alone for data-intensive sciences. Active storage, active disks, and smart disks have gained increasing attention in recent years [17, 18]. Active storage leverages the computing capability of storage nodes and performs certain computation to reduce the bandwidth requirement between storage and compute nodes. Active disks and smart disks integrate a processing unit within disk storage devices and offload computations to embedded processing unit. However, these architecture improvements are designed to explore either the idle computing power of storage nodes or an embedded processor, and have limited computation-offloading capability. Blue Gene Active Storage [19] is a recent implementation of active storage prototype on the IBM Blue Gene platform. The Oracle Exadata [20] is another active storage-like system but focuses on scan-intensive database queries (read operations). The decoupled HPC system architecture provides a more powerful platform for the same purpose [21]. In addition, it handles both read and write operations. I/O forwarding (both hardware and software solutions) [22, 23] and data shipping [24] provide approaches to offloading I/O requests to dedicated nodes, aggregating the requests, and carrying out them on behalf of compute nodes. The data nodes in the decoupled system architecture can carry all these functions.

Current parallel programming models are designed for computation-intensive applications. These programming models include Message Passing Interface (MPI), Global Arrays, OpenMP, Unified Parallel C, Chapel, X10, Co-array Fortran, and data parallel programming models such as High Performance Fortran (HPF). These programming models primarily focus on the memory abstractions and communication mechanism among processes. I/O is treated as a peripheral activity and often a separate phase in these programming models, which is often achieved through a subset of interfaces such as MPI-IO [25]. Advanced I/O libraries, such as Hierarchical Data Format (HDF), Parallel netCDF (PnetCDF), and Adaptable IO System (ADIOS) [26], provide high-level abstractions, map the abstractions onto I/O in one way or another [27, 28], and complement parallel programming models in managing data access activities. The MapReduce programming model [10, 29, 30] is an instant hit and has been proven effective for many data-intensive applications. The MapReduce model, however, is typically layered on top of distributed file systems and is not designed for HPC semantics. It requires specific Map and Reduce abstractions as well [10, 29]. The decoupled HPC system architecture is studied for general HPC applications.

There have been significant amount of research efforts in optimizing data-access performance using runtime libraries. Abbasi et. al. proposed a DataStager framework with data staging services that move output data to dedicated staging or I/O nodes prior to storage, which has been proven effective

in reducing the I/O overheads and interferences on compute nodes [31]. Zheng et. al. proposed a preparatory data analytics approach to preparing and characterizing scientific data when generated (e.g. data reorganization and metadata annotation) to speedup subsequent data access [32]. These approaches have shown considerable performance improvement with dedicated output staging services and preparatory analysis. A decoupled HPC system architecture studied in this research leverages dedicated nodes as well. These data nodes can provide buffering or staging too, but more importantly on data reduction. The notion of data processing nodes is a revisit of HPC system architecture to provide balanced computational and data-access capability. The decoupled HPC system architecture considers to address the fundamental architecture issue for data-intensive sciences.

V. Conclusion

Many scientific computing applications have become increasingly data-intensive. These applications have brought up an important question to the HPC research and development community - how to efficiently support data-intensive sciences with HPC systems, while conventional HPC systems are designed for computation-intensive applications. The massive amount of data movement and long access delay can significantly limit the productivity of data-intensive scientific applications.

In this paper, we present our research study trying to answer the above question with revisiting HPC system architecture. We study a decoupled HPC system architecture for scientific big data applications. The decoupled architecture builds separate data processing nodes and compute nodes, with computation-intensive and data-intensive operations mapped to compute nodes and data processing nodes respectively. The data processing nodes and compute nodes collectively provide a balanced system design for data-intensive applications. We have presented modeling and analyses to study the potential. The result has shown a promising potential of such a decoupled HPC system architecture. We were able to draw important conclusions for HPC system design and development, and these conclusions can guide the configuration and deployment of future HPC systems for solving data-intensive scientific problems. While this study is one of steps of we are trying to develop better HPC solutions for scientific big data applications, the current results are encouraging. Given the growing importance of supporting data-intensive sciences, such a decoupled HPC system architecture can have an impact. It can potentially guide building exascale HPC systems as well to better support data-intensive sciences.

VI. Acknowledgment

This research is sponsored in part by the National Science Foundation under grant CNS-1162540, CNS-1162488, CNS-1161507, and CNS-1338078. The authors acknowledge the High Performance Computing Center (HPCC) at Texas Tech University at Lubbock for providing HPC resources that have contributed to the research results reported within this paper. URL: http://www.hpcc.ttu.edu. The authors would also like to acknowledge anonymous reviewers for their suggestions that improved this research study.

References

[1] J. Dongarra, P. H. Beckman, and T. M. etc., "The International Exascale Software Project Roadmap," *IJHPCA*, vol. 25, no. 1, pp. 3–60, 2011.

[2] V. Sarkar, S. Amarasinghe, and D. C. etc., "ExaScale Software Study : Software Challenges in Extreme Scale Systems," *ExaScale Computing Study*, pp. 1–159, 2009.

[3] "DOE Innovative and Novel Computational Impact on Theory and Experiment Program," http://hpc.science.doe.gov/.

[4] R. Ross, R. Latham, M. Unangst, and B. Welch, "Paralell I/O in Practice," in *Tutorial in the ACM/IEEE SC'09 Conference*, 2009.

[5] "Global Cloud Resolving Model (GCRM)," https://svn.pnl.gov/gcrm.

[6] V. R. Borkar, M. J. Carey, and C. Li, "Big data platforms: what's next?" *ACM Crossroads*, vol. 19, no. 1, pp. 44–49, 2012.

[7] L. Liu, "Computing infrastructure for big data processing," *Frontiers of Computer Science*, vol. 7, no. 2, pp. 165–170, 2013.

[8] T. Condie, P. Mineiro, N. Polyzotis, and M. Weimer, "Machine learning for big data," in *SIGMOD Conference*, 2013, pp. 939–942.

[9] "Community Earth System Model," http://www.cesm.ucar.edu.

[10] J. Dean and S. Ghemawat, "MapReduce: Simplied Data Processing on Large Clusters," in *Operating Systems Design and Implementation*, 2004, pp. 137–150.

[11] "IOSIG Project," http://www.cs.iit.edu/~scs/iosig.

[12] J. He, J. Bent, A. Torres, G. Grider, G. A. Gibson, C. Maltzahn, and X.-H. Sun, "I/O Acceleration with Pattern Detection," in *HPDC*, 2013, pp. 25–36.

[13] Y. Chen, S. Byna, X.-H. Sun, R. Thakur, and W. Gropp, "Hiding I/O Latency with Pre-execution Prefetching for Parallel Applications," in *Proc. of the 2008 ACM/IEEE conference on Supercomputing*, ser. SC '08, 2008, pp. 40:1–40:10.

[14] F. Chen, D. A. Koufaty, and X. Zhang, "Hystor: Making the Best Use of Solid State Drives in High Performance Storage Systems," in *ICS*, 2011, pp. 22–32.

[15] S. Chen, P. B. Gibbons, and S. Nath, "Rethinking database algorithms for phase change memory," in *CIDR*, 2011, pp. 21–31.

[16] X. Dong and Y. Xie, "AdaMS: Adaptive MLC/SLC Phase-change Memory Design for File Storage," in *ASP-DAC*, 2011, pp. 31–36.

[17] Y. Xie, K.-K. Muniswamy-Reddy, and D. F. etc., "Design and Evaluation of Oasis: An Active Storage Framework based on T10 OSD Standard," in *MSST*, 2011.

[18] S. W. Son, S. Lang, and P. e. Carns, "Enabling Active Storage on Parallel I/O Software Stacks," in *Proceedings of the 2010 IEEE 26th Symposium on Mass Storage Systems and Technologies (MSST)*, ser. MSST '10, 2010, pp. 1–12.

[19] B. G. Fitch, A. Rayshubskiy, M. P. T.J. Chris Ward, B. Metzler, H. J. Schick, B. Krill, P. Morjan, and R. S. Germain, "Blue Gene Active Storage," in *HEC FSIO R&D Workshop '10*, 2010.

[20] "Oracle Exadata Database Machine," http://www.oracle.com/us/products/database/exadata/database-machine-x3-8/overview.

[21] Y. Chen, C. Chen, X.-H. Sun, W. D. Gropp, and R. Thakur, "A Decoupled Execution Paradigm for Data-Intensive High-End Computing," in *In the Proc. of the IEEE International Conference on Cluster Computing 2012 (Cluster'12)*, 2012.

[22] N. Ali, P. H. Carns, and K. I. etc., "Scalable I/O Forwarding Framework for High-performance Computing Systems," in *CLUSTER*, 2009.

[23] K. Iskra, J. W. Romein, K. Yoshii, and P. Beckman, "ZOID: I/O-forwarding Infrastructure for Petascale Architectures," in *PPoPP*, 2008.

[24] F. Schmuck and R. Haskin, "GPFS: A Shared-Disk File System for Large Computing Clusters," in *Proceedings of the 1st USENIX Conference on File and Storage Technologies*, 2002.

[25] R. Thakur, R. Ross, E. Lusk, and W. Gropp, "Users Guide for ROMIO: A High-Performance, Portable MPI-IO Implementation," *Mathematics and Computer Science Division*, 1997.

[26] J. Lofstead, M. Polte, G. Gibson, S. Klasky, K. Schwan, R. Oldfield, M. Wolf, and Q. Liu, "Six Degrees of Scientific Data: Reading Patterns for Extreme Scale Science IO," in *Proc. of HPDC*, 2011, pp. 49–60.

[27] H. Abbasi, G. Eisenhauer, M. Wolf, K. Schwan, and S. Klasky, "Just in time: Adding Value to the IO Pipelines of High Performance Applications with JITStaging," in *HPDC*, 2011, pp. 27–36.

[28] P. Widener, M. Wolf, H. Abbasi, S. Mcmanus, M. Payne, M. Barrick, J. Pulikottil, P. Bridges, and K. Schwan, "Exploiting Latent I/O Asynchrony in Petascale Science Applications," *IJHPCA*, vol. 25, pp. 161–179, 2011.

[29] S. Sehrish, G. Mackey, J. Wang, and J. Bent, "MRAP: a novel MapReduce-based framework to support HPC analytics applications with access patterns," in *IEEE International Symposium on High Performance Distributed Computing*, 2010, pp. 107–118.

[30] R. Grover and M. J. Carey, "Extending map-reduce for efficient predicate-based sampling," in *ICDE*, 2012, pp. 486–497.

[31] H. Abbasi, M. Wolf, and G. e. Eisenhauer, "DataStager: Scalable Data Staging Services for Petascale Applications," in *HPDC*, 2009.

[32] F. Zheng, H. Abbasi, and C. D. etc., "Predata - preparatory data analytics on peta-scale machines," in *IPDPS*, 2010, pp. 1–12.