Statement of Research
Aniruddha S. Gokhale (a.gokhale@vanderbilt.edu)
Associate Professor, Dept. of EECS
Vanderbilt University, Nashville, TN 37235
July 29, 2017

1 Executive Summary
At its core, my research blends Software Engineering principles with Systems Research. In this context currently I am focusing on developing novel algorithms and software-defined techniques to solve a myriad of distributed systems challenges in realizing resilient cyber physical systems. Specifically, the contours of my current research involve addressing challenges in mobile and edge cloud computing, which in turn includes (a) dynamic resource management based on dynamic system model learning, simulation and system actuation, (b) reconciling real-time stream processing – which is required for the distributed Big Data analytics, with data-centric publish/subscribe systems – which is required for information-centric networking, and (c) algorithms and mechanisms for software-defined, resilient infrastructures – which are required to manage and safeguard system resources and applications. An outreach aspect of my current work focuses on using cloud computing technologies for STEM education using real-world problems from my current research as challenge problems for STEM education and collaborative learning. My ongoing and planned research, as presented in this document, has been informed by emerging trends and insights gained from work to date.

2 Emerging Trends and Research Vision
The proliferation of smart, mobile endpoints including sensors and smartphones that form the genesis of the Internet of Things (IoT) paradigm, and the ensuing massive amounts of Machine-to-Machine (M2M) communications, has created a myriad of challenges that manifest in the next-generation of cyber Physical Systems (CPS), such as Smart and Connected Cities, Smart Grids and Smart Manufacturing. Addressing these challenges forms my research agenda over the next five years.

To understand these challenges, consider how these CPS are increasingly supporting device-to-cloud as well as cloud-to-cloud communication patterns for scalable data analytics and fault tolerance requirements [41, 45]. Thus, the communication networks for these CPS must interconnect a variety of smart objects that are designed for different hardware platforms, support large-scale deployment of IoT devices, and handle a massive number of generated events. Interconnecting heterogeneous smart objects over the Internet is, however, a very difficult goal to achieve because these CPS applications operate in diverse environments – often remote and hostile – and involve different functional and quality of service (QoS) requirements. Provisioning and sharing the communication links across the CPS for the diverse needs of the different traffic flows remains an unresolved problem.

At the same time, due both to the resource-constrained nature of the system (e.g., available communication bandwidth and battery power) and low latency requirements of many of the CPS applications (e.g., driver notification in a smart transportation system), the data analytics (e.g., analyzing traffic congestion data in a segment of a road network) cannot always be performed in a centralized cloud, which may be many
network hops away from the source of the data and can cause substantial delay in the results propagating back. Consequently, the data analytics must be performed in a distributed manner and often closer to the source. The emergence of cloudlets [123], which are akin to micro data centers and are closer to the source, as well as the increasingly powerful edge devices make edge/cloudlet computing appealing to address these challenges.

Despite this promise, the outcomes of the individualized and localized analytics may not be sufficient to obtain an accurate operating picture of the overall system, which is needed for the effective management and resilience of the system through a range of actions, such as dynamic management and scheduling of resources, and reconfiguration and redeployment of applications. Consequently, information-centric networking solutions that process the deluge of information in real-time and in a distributed manner opportunistically at distributed resources ranging from the edge to the cloud [134], and which can disseminate the outcomes to interested parties over wide area networks [6] are needed.

My research agenda is addressing these intertwined set of challenges holistically through the following primary set of synergistic research efforts.\(^1\)

- **Distributed, real-time stream processing blended with wide-area publish/subscribe (pub/sub):** New research directions that will realize WAN-scale pub/sub for CPS that provide control over the communication links, and which enable both edge- and cloud-based analytics is needed. In this context, we focus on data-centric pub/sub since it provides the desired semantics for event-based interactions ensuring scalable and decoupled data sharing mechanisms between communicating peers within CPS [6].

  Coupled with this information dissemination focus, we are investigating mechanisms to execute real-time stream processing tasks for model learning opportunistically across the spectrum of resources ranging from the edge to the cloud. In doing so, our aim is to reconcile the real-time needs of stream processing with that of real-time dissemination. Together, this approach will help to develop a global system model that can be used for system management and resilience [82].

- **Dynamic resource management:** To enable any of the earlier research foci, effective dynamic resource management across the spectrum of resources is required. Such a system-level resource management requires effective models of the system, which must be learned online and subsequently used to control and schedule the resources [25, 29, 134]. This dimension of the problem is different from the application-level analytics. To that end, the software-defined approaches that separate the control plane from the data plane are investigated so that system-level activities including analytics and dissemination are performed at the control plane such that any changes to the algorithms and policies can impact only the control plane while not impacting the data plane where the applications reside [65, 106, 62, 63, 105].

- **Performance optimizations and system resilience:** A consistent theme of my research right from my doctoral research has been about performance optimization and resilience. Proper and principled implementation of the devised algorithms is necessary to realize elegant and extensible implementations that make optimal use of resources; this issue is all the more important for resource-constrained parts of the CPS we consider in our research [27, 87, 23]. Similarly, system resilience is another requirement for the CPS where human-driven control may not be feasible at all times. Thus, my existing collaborative research efforts are exploring the use of goal-oriented, autonomous system (re)configuration and (re)deployment to make them resilient [112, 113, 111].

\(^1\)Individual problem areas of research and resulting contributions are not discussed in this document; instead these can be obtained from my publications listed in my CV.
These research ideas are informing my agenda for education and outreach. For example, conducting these research investigations in the context of current projects, e.g., NSF US Ignite’s software-defined networking project, Siemens’ Edge computing project, AFOSR DDDAS’s dynamic model learning project, and Vanderbilt TIPS’ Smart cities project, are providing me with real-world scenarios and challenge problems that I am using to further improve the web-based STEM learning environment called $C^3STEM$ that we have built as part of an NSF project and have been using to conduct user studies with high school students from the Nashville metropolitan public schools [36, 133, 37, 28]. Moreover, the harder challenge problems also serve as interesting project topics for my Cloud Computing (CS 4287/5287) and Distributed Systems (CS 6381) courses. These also will form the basis of a Distributed Systems Specialization MOOC that I plan to co-offer on the Coursera platform in Fall 2018.

The rest of this document presents my research contributions in reverse chronological order. This order was deemed necessary since the most recent research provides the most compelling arguments for my current and planned research directions. Section 3 presents insights gained from research conducted post promotion and tenure; Section 4 presents a summary of my research during my years as a tenure track faculty; Section 5 presents a summary of my industrial research; Section 6 presents a summary of my doctoral research; and finally Section 7 discusses my research plan including current and planned collaborations within and outside of Vanderbilt University.

### 3 Contributions and Insights from Research Conducted as Associate Professor (Sept 2010 – present)

The research I have been involved in after being tenured and promoted to Associate Professor has been instrumental in shaping up the ideas for my ongoing and planned research agenda. This recently concluded research can be classified along the following general themes and dimensions, all applied to supporting various properties of cyber physical systems: (a) Resource Management and Autoscaling for the Cloud infrastructure, (b) Software-defined Networking solutions to control networking resources, (c) Parallel dataflow models for in-network data analytics and dissemination, (d) Middleware specialization, and (c) Deployment and configuration heuristics. A major portion of my contributions during this time were funded by my NSF CAREER award, which I had secured a year prior to tenure, by AFOSR DDDAS award, and recently also by the NSF US Ignite award.

#### 3.1 Cloud Infrastructure Resource Management to Support CPS Applications

Contemporary and next-generation of CPS need to process significant volumes of data, which are generated by a heterogeneous set of sources, e.g., mobile devices, social media, and a number of the sensors. In the next five years, it is expected that mobile traffic will have grown thirteen times more than the existing mobile traffic and there will be three times more connected devices than the number of people on the Earth [74]. Similarly, scientific experiments such as CERN also generate enormous amounts of data estimated to be about twenty-five petabytes in a year [99]. With the emergence of the IoT paradigm, billions of data points are generated and as a result, the volume of this data is getting even larger.

All of this generated data must be processed to extract useful features out of it. This growing, massive amounts of data require more storage and compute resources, which is ultimately provided by the data centers throughout the world and the cloud computing infrastructure. As more and more applications are created, the cloud computing in general and data center in particular have become critical for many projects, enterprises, and research communities. Hence, it will continue to play a crucial role in delivering a variety of services.

Despite the fact that there is a significant momentum towards moving to the cloud, a variety of issues still exist in utilizing the cloud to its fullest potential. For example, energy efficiency, capacity planning,
performance management, disaster management, and security are a few major concerns faced by cloud service providers (CSPs) among others. The energy consumption of data centers worldwide has reached staggering proportions and this trend will further continue. Moreover, diesel power generators, due to power outages in data centers and power plants, emit millions of tons of carbon [145, 84]. Thus, CSPs must address energy efficiency issues for data centers. At the same time, meeting the timeliness and reliability of CPS applications executing in the cloud is paramount. To that end, my research has addressed the following challenges in the context of enabling the cloud platforms for CPS applications.

3.1.1 Challenge 1: Autonomous and Dynamic Scheduler Reconfiguration

At the virtualization layer of a data center, hypervisors have a scheduling mechanism to deal with sharing CPU resources among the virtual machines (VMs) and executing the workloads in the VMs. Relying on default values, manually tuning the scheduler’s parameters by following known configuration patterns, using generally accepted rules, and adopting trial-and-error approach, are common practices among the system administrators of the cloud data center. However, these approaches are not effective and efficient, particularly when dealing with dynamically changing workloads on the host machines and varied CPU resource utilizations. Moreover, these non-scientific approaches do not consider the resource overbooking ratios for resource management. Furthermore, often these manual decisions are made offline, which invariably cannot consider the overall system dynamics leading to poor system performance. Therefore, an online, autonomous, and self-tuning system for scheduler configuration is desired.

To address this challenge, we have developed iTune [27], which is a middleware that optimizes the Xen hypervisor’s scheduler configuration parameters autonomously through a three phase design workflow comprising: (1) Discoverer, which monitors and saves the resource usage history of the host machines and groups set of related host machine workload, (2) Optimizer, where optimum Xen scheduler configuration parameters for each workload cluster is explored by employing a simulated annealing machine learning algorithm, and (3) Observer, where iTune monitors the resource usage of host machines online, classifies them into one of the categories found in the Discoverer phase, and loads the optimum scheduler parameters determined in the Optimizer phase.

3.1.2 Challenge 2: Resource-Overbooking to Support Soft Real-time Applications

Under-utilization, wastage of resources, and inefficient energy consumption are among the traditional issues of crucial importance to data centers. CSPs often overbook their resources by utilizing the tools in the cloud management layer. Overbooking is an attractive strategy to CSPs because it helps to reduce energy consumption and increase resource utilization in the data center by packing more user jobs in a fewer number of resources while improving their profits. Overbooking becomes feasible because cloud users tend to overestimate their resource requirements, utilizing only a fraction of the allocated resources. Without overbooking, resources in a data center will otherwise remain under-utilized.

Resource overbooking ratios are generally determined sporadically by analyzing the historic resource usage of workloads or following the best practices. Unfortunately, governing cloud resources in this manner may be detrimental and catastrophic to soft real-time applications running in the cloud. To make systematic and online determination of overbooking ratios such that the quality of service needs of soft real-time systems can be met while still benefiting from overbooking, there is a need for more efficient, effective, and intelligent approaches to overbooking that will ensure good performance for soft real-time applications yet prevent under utilization and also save energy costs.

To address this challenge, we have developed iOverbook [25], which is an overbooking strategy that uses a machine learning approach to make systematic and online determination of overbooking ratios such that the quality of service needs of soft real-time systems can be met while still benefiting from overbooking. Specifically, iOverbook utilizes historic data of tasks and host machines in the cloud to extract their
resource usage patterns and predict future resource usage along with the expected mean performance of host machines. To evaluate our approach, we have used a large usage trace made available by Google of one of its production data centers.

3.1.3 Challenge 3: Performance Interference Effects on Application Performance

Although resource overbooking in the cloud is standard practice, it can lead to performance interference and anomalies among the VMs hosted on the physical resources, causing performance unpredictability for soft real-time applications hosted in the VMs. Such unpredictability may be detrimental to the performance of CPS applications that are controlled by the models executing in the cloud infrastructure. Moreover, resource overbooking can propagate and trigger faults in other VMs, which is also not acceptable to CPS applications. To address these problems and because workloads of the VMs may change at run time, virtual machine migration between physical host machines and data centers is the generally accepted mechanism.

Choosing the right set of target physical host machines for VM migration decisions plays a critical role in determining the performance and interference effects post migration. Analyzing the performance anomalies that might occur and predicting performance interference and fault before a VM is deployed or migrated on the physical host machines is thus desired and vital for soft-real-time applications.

To address these issues, we have developed iSensitive [29], which is a machine learning-based middleware providing an online placement solution where the system is trained using events and lifecycle of a publicly available trace of a large data center owned by Google. Our approach first classifies the VMs based on their historic mean CPU, memory usage, and network usage features. Subsequently, it learns the best patterns of collocating the classified VMs by employing machine learning techniques. These extracted patterns document the lowest performance interference level on the specified host machines making them amenable to hosting applications while still allowing resource overbooking.

3.1.4 Challenge 4: Power- and Performance-Aware Virtual Machine Placement

Virtual machines are migrated from one physical host machine to another one in the same data center or across the data centers located in different locations due to fault tolerance, balance workload, application performance management concerns, and eliminate hotspots. Apart from the performance interference aspects described above, power and performance trade-offs are also critical and challenging issues faced by CSPs while managing their data centers. On the one hand, CSPs strive to reduce power consumption of their data centers to not only decrease their energy costs but also to reduce adverse impact on the environment. On the other hand, CSPs must deliver performance expected by the applications hosted in their cloud data centers in accordance with predefined Service Level Objective (SLOs). Not doing so will lead to loss of customers and thereby major revenue losses for the CSPs.

Power management and performance assurance are conflicting objectives, particularly in the context of multi-tenant cloud systems where multiple VMs may be hosted on a single physical server. The problem becomes even harder when soft real-time applications are hosted in these VMs. Solutions to address the virtual machine placement decisions exist. Bin packing heuristics such as first-fit, best-fit, and next-fit are common practices used by cloud management platforms (e.g., OpenNebula, OpenStack, etc.) to deploy VMs in the cloud. However, these solutions do not consider application performance and energy efficiency. To address the aforementioned issues, a power and performance-aware virtual machine placement algorithm is desired.

To address this challenge, we have developed iPlace [26], which is a middleware providing an intelligent and tunable power- and performance-aware VM placement capability. The placement strategy is based on a two-level artificial neural network, which predicts (1) CPU usage at the first level, and (2) power consumption and performance of a host machine at the second level that uses the predicted CPU usage. The
placement decision (i.e., aptly suited host machine for the VM being deployed) is determined by making the appropriate trade-offs between predicted power and performance values of a host machine.

### 3.1.5 Challenge 5: Supporting Stochastic Hybrid Models of DDDAS Applications

With the advent of the Internet of Things (IoT) paradigm [13], there is no dearth of collected data. When coupled with technology advances in mobile computing and edge devices, users are expecting newer and different kinds of services that will help them in their daily lives. For example, users may want to determine appropriate temperature settings for their homes such that their energy consumption and energy bills are kept low yet they have comfortable conditions in their homes. Other examples include estimating traffic congestion in a specific part of a city on a special events day.

Deploying these services in-house is unrealistic for the users since the models of these systems are quite complex to develop. Some models may be stochastic in nature, which require a large number of compute-intensive executions of the models to obtain outcomes that are within a desired statistical confidence interval. Other kinds of simulation models require running a large number of simulation instances with different parameters. Irrespective of the simulation model, individual users and even small businesses cannot be expected to acquire the needed resources in-house. Cloud computing then becomes an attractive option to host such services particularly because hosting high performance and real-time applications in the cloud is gaining traction [3, 91].

Running these simulations sequentially is not a viable option as user expectations in terms of response times have to be met. Hence there is a need for a simulation platform where a large number of independent simulation instances can be executed in parallel and the number of such simulations can vary elastically to satisfy specified confidence intervals for the results.

To that end we have architected a cloud-based solution comprising resource management algorithms and middleware called Simulation-as-a-Service (SIMaaS) [135]. SIMaaS can elastically and on-demand execute the multiple different simulation trajectories of the simulation models in parallel, and perform aggregation such as stochastic model checking (SMC) to obtain results within a desired confidence interval. SIMaaS leverages Linux container [88]-based infrastructure, which has low runtime overhead, higher level of resource sharing, and very low setup and tear down costs. It provides a resource management algorithm, that reduces the cost to the service provider and enhances the parallelization of the simulation jobs by fanning out more instances until the deadline is met while simultaneously auto-tuning itself based on the feedback. The SIMaaS middleware intelligently generates different configurations for experimentation, and intelligently schedules the simulations on the Linux container-based cloud to minimize cost while enforcing the deadlines.

### 3.2 Software-defined Networking Approaches to Control Communication Resources

Software-Defined Networking (SDN) has emerged as a new intelligent architecture for network programmability. It moves the control plane outside the switches to enable external centralized control of data through a logical software entity called controller. The controller offers northbound interfaces to network applications that provide higher level abstractions to program various network-level services and applications. also uses southbound interfaces to communicate with network devices. One complementary technology to SDN called Network Function Virtualization (NFV) has the potential to dramatically impact future networking by providing techniques to refactor the architecture of legacy networks by virtualizing as many network functions as possible. NFV advocates the virtualization of network functions as software modules running on standardized IT infrastructure (like commercial off-the-shelf servers), which can be assembled and/or chained to create services.

Beyond traditional data centers where SDN is most applicable, the emergence of the IoT paradigm has given rise to many challenges. From a communications perspective, a high-level architecture of IoT
systems is typically composed of three domains: Device Domain, Network Domain and the Application Domain. In the device domain, the device provides direct connectivity to the network domain via access networks, which may include limited range PAN technologies such as Bluetooth, ZigBee, etc., or via a gateway that acts as a network proxy for the network domain. Such a gateway must be flexible enough to efficiently manage available resources, QoS, security, as well as multimedia data exchange. These gateway concepts are prevalent in home ADSL models and WiFi access points found in cyber cafes and wireless hotspots. Because IoT systems integrate heterogeneous smart objects, the design of the gateway is quite different because it must not require each IoT subnetwork to have its own gateway. Thus, a convergent architecture towards a unique solution that integrates traffic incoming from heterogeneous smart devices should be designed.

Furthermore, as smart objects are resource- and energy-constrained, the gateway should be aware of the context of each process being managed. It should also employ intelligent routing protocols and caching techniques to route the traffic across the less constrained paths. The network domain includes different access networks, which provide connectivity through diverse technologies, such as xSDL, Satellite, etc, to devices and/or gateways. It also provides connectivity to the core network that includes heterogeneous and multi-technology connectivity, such as 3GPP, TISPAN, and LTE-A. Finally, the application domain includes the IoT applications and server/cloud infrastructures. The latter have to share their content, possibly back them up to other devices, analytic programs, and/or people who need to monitor real-time response to events. They also include service capabilities, which provide functions shared between different applications through open, high-level abstractions and interfaces that hide the specificities of the underlying networks.

My recent NSF US Ignite award is supporting my research investigations in the area of software defined networking as applied to addressing the myriad of challenges in traditional data centers as well as in wireless IoT environments. This dimension of research is still evolving and the following contributions have been made since early 2015.

- Articulated several different open challenges in SDN for both wired and wireless networks [65].
- Proposed an approach for bootstrapping the controller functionality in a project called InitSDN [106].
- Proposed an approach to use data-centric publish/subscribe as the messaging layer for the SDN northbound communications for IoT applications [62].
- Proposal a design of low rate wireless personal access networks (LR-WPAN) IoT Systems using SDN [64].
- Proposed an approach to use data-centric publish/subscribe as the messaging layer for a scalable southbound layer in SDN [63].
- Proposed an approach for data dissemination in wireless mesh smart grids using SDN [?].
- Proposed an approach to realize adaptive SDN-based multicast for group communications in data centers (paper in submission [107]).
- Proposed an approach for SDN-enabled wireless fog network management (paper in submission [66]).
- Proposed an approach for enabling SDN approach to be used in wireless mesh networks using a three-stage routing protocol [105].

3.3 Parallel Dataflow Models for In-Network Analytics and Dissemination

Critical CPS, such as smart-grids, intelligent transportation systems, advanced manufacturing, health-care tele-monitoring, etc, share several key cross-cutting aspects. First, they are often large-scale, distributed
systems comprising several, potentially mobile, publishers of information that produce large volumes of asynchronous events. Second, the resulting unbounded asynchronous streams of data must be combined with one-another and with historical data and analyzed in a responsive manner. While doing so, the distributed set of resources and inherent parallelism in the system must be effectively utilized. Third, the analyzed information must be transmitted downstream to a heterogeneous set of subscribers. In essence, these critical systems can be understood as a distributed asynchronous dataflow. The key challenge lies in developing a dataflow-oriented programming model and a middleware technology that can address both distribution and asynchronous processing requirements adequately.

The distribution aspects of dataflow-oriented systems can be handled sufficiently by data-centric publish/subscribe (pub/sub) technologies [40], such as Object Management Group (OMG)’s Data Distribution Service (DDS) [100]. DDS is an event-driven publish-subscribe middleware that promotes asynchrony and loose-coupling between data publishers and subscribers which are decoupled with respect to (1) time (i.e., they need not be present at the same time), (2) space (i.e., they may be located anywhere), (3) flow (i.e., data publishers must offer equivalent or better quality-of-service (QoS) than required by data subscribers), (4) behavior (i.e., business logic independent), (5) platforms, and (6) programming languages.

In fact, as specified by the Reactive Manifesto [2], event-driven design is a pre-requisite for building systems that are reactive, i.e. readily responsive to incoming data, user interaction events, failures and load variations- traits which are desirable of critical systems. Moreover, asynchronous event-based architectures unify scaling up (e.g., via multiple cores) and scaling out (e.g., via distributed compute nodes) while deferring the choice of the scalability mechanism at deployment-time without hiding the network from the programming model. Hence, the asynchronous and event-driven programming model offered by DDS makes it particularly well-suited for demanding systems.

However, the data processing aspects, which are local to the individual stages of a distributed dataflow, are often not implemented as a dataflow due to lack of sufficient composability and generality in the application programming interface (API) of the pub/sub middleware. DDS offers various ways to receive data such as, listener callbacks for push-based notification, read/take functions for polling, waitset and read-condition to receive data from several entities at a time, and query-conditions to enable application-specific filtering and demultiplexing. These primitives, however, are designed for data and meta-data delivery as opposed to processing. Further, the lack of proper abstractions forces programmers to develop event-driven applications using the observer pattern—disadvantages of which are well documented [90].

To that end we have investigated the use of Functional Reactive Programming (FRP) for DDS applications [82]. FRP has emerged [136] as a promising new way to create scalable reactive applications and has already shown its potential in a number of domains including robotics [108, 146], animation [39], HD video streaming [1], and responsive user interfaces [92]. These domains are reactive because they require interaction with a wide range of inputs ranging from keyboards to machinery. FRP is a declarative approach to system development wherein program specification amounts to “what” as opposed “how”. Such a program description can be viewed as a data-flow [30] system where the state and control flow are hidden from the programmers. FRP offers high-level abstractions that avoid the verbosity that is commonly observed in callback-based techniques. Furthermore, FRP avoids shared mutable state at the application-level, which is instrumental for multicore scalability.

### 3.4 Middleware Specialization

High confidence software, which includes the middleware platforms, is a crucial design consideration for cyber physical systems. Developing and deploying high confidence software for cyber physical systems is fraught with multiple challenges. Addressing these challenges and realizing the goals of CPS has been a major research focus for me since my promotion and tenure.

Due to the varying constraints of both the operating environment of CPS and constraints on the re-
sources, CPS applications often do not require all the features provided by contemporary, general-purpose middleware. A lack of support for selective use of features forces applications to incur the overhead of linking in all the capabilities, which degrades QoS and increases memory footprint. Second, not only does general-purpose middleware lack out-of-the-box support for diverse and rich domain-specific properties (e.g., a custom fault masking and recovery capability), but also lack modular extensibility for both domain-specific and -independent features.

A promising approach to address this problem is to specialize general-purpose middleware (i.e., prune unwanted features, add and customize the necessary ones). The research I conducted as part of my NSF CAREER research involved (a) identifying and exploiting the algebraic structure of middleware that helps map the specialization problem into a feature-oriented software development problem; (b) developing a new theory for feature composition, refactoring, and interactions across the lifecycle stages of applications; and (c) automating these specializations through well-defined processes.

The Intellectual Merits of this research included novel software engineering approaches to specializing middleware [33, 35, 34, 142], model-based techniques [79, 24, 5, 132, 7], fault tolerance algorithms [140, 141, 32] and resource management algorithms [8, 4, 42, 149, 9, 10].

The broader impact from this research included (a) stimulating new directions in middleware design with use cases in virtualization environments; (b) documentation of specialization patterns, which help to reduce software maintenance and configuration challenges in product lines; (c) enhanced developer productivity and system correctness; and (d) education and technology transfer. We created an open source tool suite called GAMMA (Generative Aspects for the Manipulation of Middleware Architectures).

### 3.5 Deployment and Configuration Heuristics

Determining how to deploy software to hardware in CPS is a challenging problem. Developers must ensure that each software component is provided sufficient processing time to meet any real-time scheduling constraints. Resource constraints, such as total available memory on each processor, must also be respected. Finally, components can have complex placement or co-location constraints, such as requiring specific software components to be deployed to processors due to physical constraints imposed by the domain.

An NSF CNS/SHS award supported this research. The Intellectual Merit of this work lies in the investigations of formalisms and techniques for effective deployment and configuration of distributed, real-time and embedded systems. To that end we have devised mechanisms both as mathematical formulations as well as middleware artifacts, such as optimizations for runtime deployment and configuration. One such approach resulted in new hybrid algorithmic techniques for deployment of CPS. It comprises new bin-packing based deployment algorithms that accept co-location, scheduling, and resource constraints. It contained hybrid bin-packing/evolutionary algorithms for handling network constraints and non-linear optimization. The complete research resulted in a number of publications. [15, 122, 121, 131, 11, 120, 110, 102, 103, 144, 38, 104, 109, 12, 24, 8, 4, 114, 101].

The broader impact from this research include less expensive and correct deployment of and energy efficient CPS. By packing software more tightly onto hardware, fewer hardware resources can be used. For example, roughly four pounds of cooling, power supply, and other supporting hardware are needed for each pound of processing hardware in a plane. Reducing hardware not only decreases CPS cost, but also can facilitate increased ranges for planes/cars, decreased fuel and power consumption, and reduced waste.

### 4 Foundational Research Activities at Vanderbilt University (2002-2009)

My research activities at Vanderbilt University prior to obtaining tenure first started as a research scientist in the Institute for Software Integrated Systems (ISIS) in Jan 2002 and subsequently as a tenure-track assistant professor (starting Sept 2003) in the Department of Electrical Engineering and Computer Science.
has investigated solutions to address the deployment and configuration, and QoS management challenges in component-based DRE systems.

4.1 Contributions to Science and Broader Impact: Model-driven Middleware

To better enable the use of component middleware and rapid application composition with assured QoS in the context of DRE systems, my research has formulated the *Model-driven Middleware* paradigm, which combines the strengths of model-driven engineering (MDE) [125] with that of QoS-enabled component middleware [147]. The key research contributions and the challenges they resolve are described below:

1. **Technology for effectively composing DRE systems from components.** QoS-enabled component middleware enables application developers to develop individual components that can be composed together into *assemblies* that form complete DRE systems. Although this approach supports the use of “plug and play” components in DRE systems, system integrators face the daunting task of composing the right set of compatible components that will deliver the desired semantics and QoS to applications that execute in large-scale systems.

2. **Technology for configuring component middleware.** In QoS-enabled component middleware frameworks, application components and the underlying component middleware services can have a large number of attributes and parameters that can be configured at various stages of the development lifecycle, such as (1) *during component development*, where default values for these attributes could be specified, (2) *during application integration*, where component defaults could be overridden with domain specific defaults, and (3) *during application deployment*, where domain specific defaults are overridden based on the actual capabilities of the target system. It is hard, however, to manually ensure that all these parameters are semantically consistent throughout a large-scale DRE system.

3. **Technology for automated deployment of DRE systems on heterogeneous target platforms.** The component assemblies described above must be deployed in the distributed target environment before applications can start to run. DRE system integrators must therefore perform the complex task of mapping the individual components/assemblies onto specific nodes of the target environment. This mapping process involves ensuring semantic compatibility between the requirements of the individual components, and the capabilities of the nodes of the target environment.

4. **Technology for performance evaluation for DRE systems.** It is critical that the component assemblies, their configurations and deployment decisions be validated for conformance to their QoS requirements. We realized that there was a dearth of tools and techniques that enabled automated performance evaluation as well as provided design-time estimates of the expected performance of the overall system.

5. **Technology for survivable DRE systems.** It is essential that the operationalized DRE system continue to deliver the QoS despite fluctuations in resource availabilities and failures. This requires adaptive solutions that maintain both real-time (the performance aspect) and fault-tolerance (the availability aspect) of the DRE system.

4.2 Research Artifacts

The research artifacts stemming from my foundational research include an open source MDE tool suite called CoSMIC and a middleware infrastructure that supports both real-time and fault-tolerance.

4.2.1 CoSMIC Tool Suite

The *Component Synthesis using Model Integrated Computing* (CoSMIC) [46] tool suite is an integrated collection of model-driven environment (MDE) tools that address the key lifecycle challenges of middleware and applications in DRE systems. CoSMIC comprises the following capabilities:
I. Modeling Tools for Deployment and Configuration: CoSMIC is a collection of domain-specific modeling languages (DSMLs) [85, 18, 61], and model interpreters and transformations [77, 81, 80, 78] that enforce the principle of “correct-by-construction.”. CoSMIC [46, 57, 22, 19, 20], provides a model-driven, generative programming approach [47] to resolve tangled QoS concerns [21] of middleware-based real-time systems [57]. This includes using metamodeling [137] and domain-specific modeling languages (DSMLs) [61] to separate the application assembly, packaging, configuration and deployment concerns [143]. Generative technologies [31] are used to weave [138, 16, 139] these concerns into middleware platforms in a scalable manner [60].

II. Modeling Tools for Design-time Performance Validation: This capability can be further classified along the following three dimensions:

(a) Continuous QoS validation, which focuses on emulating the business logic of application components so that end-to-end behavior can be emulated and validated continuously throughout the application development, integration and deployment phase [71, 70, 68, 69]. Application behavior is modeled using the Component behavior Modeling Language (CBML), which is based on the mathematical formalism of Input/Output Automata [89], and the workloads are modeling using the Workload Modeling Language (WML).

(b) Validating the structure and configurations of the middleware infrastructure, where my team and I have developed model-based performance analysis solutions [58, 83, 59, 115, 116] that leverage Stochastic Rewards Nets (SRN) [93, 67] solvers to conduct automated performance analysis of different software patterns at a platform-independent level. We developed a DSML in CoSMIC called the Pattern-Oriented Software Architecture Modeling Language (POSAML) [75] to model the structure and features of middleware stacks in the form of patterns-based [130, 44] building blocks.

(c) Continuous QoS assurance via benchmarking and model checking: In [85, 86, 150] along with my collaborators I have demonstrated how benchmarking code generation techniques within CoSMIC can be applied to discover the dominant configuration options in middleware that impact system QoS. I have also used model checking [119, 73] as a means to verify the correctness of generated middleware configurations [76, 81].

4.2.2 FLARe and CORFU Middleware

My other significant research contribution pertains to the design and implementation of a real-time and fault-tolerant middleware. FLARe (Fault-tolerant Load-aware and Adaptive middlewaRe) [14, 17] provides a load-aware adaptive middleware that minimizes client response times while maximizing the system availability despite resource unavailability and failures. CORFU (COmponent Replication using Failover Units) [148, 72] leverages FLARe and makes it available to components that are replicated and treated logically as failover units.


This section outlines my research contributions as a member of technical staff (MTS) from July 1998 to Jan 2002 at Bell Laboratories, Lucent Technologies (now Nokia Bell Labs) in New Jersey.

5.1 Contributions to Science and Broader Impact: Fault Tolerant Middleware Standard

My research contributions towards addressing the need for survivable and highly available systems explored novel mechanisms to provide these assurances to applications via reusable middleware services. This research contributed to the Fault-tolerant CORBA standard.
Additional research contributions I made in the realm of customer relationship management (CRM) led to the development of efficient agent allocation algorithms that help improve the performance and reduce the costs of call center activities. This research resulted in a patent application. Moreover, it is conceivable that such algorithms can be useful in 911 emergency management call centers too.

5.2 Research Artifacts: DOORS FT CORBA

My research contributions included a prototype implementation of Fault Tolerant CORBA called DOORS that included performance optimizations [94, 95], and its standardization [96] activities within the Object Management Group (OMG). I was also involved with designing and developing fault tolerant services used in third-generation wireless call processing elements that were based on the UMTS standard. Additionally, a patent has been filed for my contributions to the customer-agent allocation algorithms in the context of networked call centers and customer relationship management.


My research focus as a doctoral student at Washington University in St. Louis was on a class of systems called the distributed, real-time and embedded (DRE) systems found in numerous domains, such as avionics mission computing, shipboard computing and industrial automation. My research focused on designing and implementing robust and optimized middleware hosting platforms for DRE systems.

6.1 Contributions to Science and Broader Impact: QoS-enabled Middleware

My doctoral research led to a key enabling technology to realize the goals of a service-oriented cyber infrastructure to host DRE systems. It realized the notion of performance-tuned middleware [124]. Specifically, my dissertation focused on devising algorithms and optimization techniques that substantially improved the performance and predictability of real-time middleware.

The research contributions included: (1) efficient protocol engines [54] based on interpretive (de)marshaling mechanisms that minimized footprint while enabling performance that rivaled compiled (de)marshaling techniques, (2) efficient request demultiplexers [52] that combined perfect hashing and active demultiplexing for constant time dispatching of messages to object implementations thereby improving predictability and scalability, and (3) generative programming [31] via interface definition language compilers [55] that used aspect-oriented generative techniques [43] to weave interpreted and compiled stubs together to achieve optimal trade-offs between footprint and performance.

A notable outcome of my doctoral research has been the documentation and codification of optimization principle patterns [56, 118], such as optimizing for the common case; eliminating gratuitous waste; replacing general purpose methods with specialized, efficient ones; precomputing values, if possible; storing redundant state to speed up expensive operations; passing information between layers; optimizing for the processor cache; and optimizing demultiplexing strategies.

6.2 Research Artifacts: The TAO ORB

My doctoral research resulted in several novel middleware benchmarking [48, 49, 51, 52] and middleware optimization techniques [54, 50, 53, 128, 55, 56, 117], which have been used to guide the implementation of a high-performance, real-time CORBA [97] Object Request Broker (ORB) [98] called TAO [126, 127, 129], which also substantially influenced the Real-time CORBA specification [97].
7 Research Management Plan

This section describes briefly how I plan to execute my research tasks to achieve both near- and longer-term objectives.

7.1 Expected Doctoral Advisee Contributions

In the following I outline the expected contributions of my current doctoral student advisees and co-advisees leading to the realization of the stated goals of my research.

1. **Shunxing Bao**, PhD expected Spring 2018; focusing on performance optimizations to Apache Hadoop ecosystem for medical image processing applications.
2. **Yogesh Barve**, PhD expected Fall 2018; focusing on reliability in high performance, distributed simulations.
3. **Anirban Bhattacharjee**, PhD expected Fall 2018; focusing on model-driven orchestration of Big Data cloud-based systems.
4. **Travis Brummett**, is a new PhD student in my group.
5. **Davut Disci**, PhD expected Spring 2019; starting to focus on Big Data storage management and technologies for STEM education.
6. **Shweta Khare**, PhD expected Fall 2018; focusing on blending real-time stream processing with data-centric publish/subscribe.
7. **Shashank Shekhar**, PhD expected Spring 2018; focusing on dynamic resource management for cloudlet/edge computing.

7.2 Research Collaborations

I am collaborating with a number of internal and external collaborators. In the following I summarize how I expect my collaborations to aid in my ongoing and future research endeavors.

1. **Drs. Xenofon Koutsoukos and Yevgeniy Vorobeychik** (ISIS/EECS, VU), with whom I am teaming up to conduct follow-on research in the DDDAS area and awaiting a decision on a follow-on grant for the DDDAS program.
2. **Drs. Abhishek Dubey** (ISIS/EECS, VU), with whom I am collaborating to further advance our work in Edge/Cloud computing. We have recently won a DURIP equipment award and will be setting up a testbed for our investigations involving joint work in Smart Cities.
3. **Dr. Bennett Landman** (EECS, VU), with whom I have teamed up for over an year and am co-advising a student, who is investigating performance optimizations in the Apache Hadoop ecosystem for medical image processing applications.
4. **Dr. Gautam Biswas** (EECS, VU), with whom I have been working for the past few years on technologies for STEM education. Currently we are funded by the NSF US Ignite program where we are exploring the use of software defined networking technologies to use cloudlets and the edge for low latency collaborative STEM learning.
5. **Drs. Abhishek Dubey, Gautam Biswas, Douglas Schmidt, Jules White** (ISIS/EECS, VU), **Dr. Hiba Baroud** (CEE, VU) and several other VU colleagues outside of the School of Engineering, with whom I have teamed up to conduct research on Smart Cities, which is funded by the Vanderbilt TIPS program.
6. **Dr. Shivakumar Sastry** (ECE, Univ of Akron), with whom I have been collaborating for the past few years in addressing a variety of problems related to advanced manufacturing.
7. **Dr. Sumant Tambe (LinkedIn) and Dr. Kyoungho An (RTI)**, with whom I have been collaborating to investigate ideas in real-time stream processing and publish/subscribe systems.

8. **Dr. Akram Hakiri (Tunisia)**, with whom I have been collaborating for the past few years on topics related to software-defined networking.

9. **Dr. Takayuki Kuroda (NEC, Japan)**, with whom I have been collaborating since his sabbatical in my lab during 2014-15 academic year on configuration automation for enterprise systems.

10. **Dr. Yogesh Simmhan (IISc, Bengaluru, India)**, with whom I had an initial kickoff meeting to forge collaborative efforts given substantial synergies in our work.

References


[44] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley, Reading, MA, 1995. 4.2.1


[79] A. Kavimandan, A. Gokhale, G. Karsai, and J. Gray. Managing the Quality of Software Product Line Architectures through Reusable Model Transformations. pages 13–22, June 2011. 3.4


