Poster Abstract: A Distributed and Resilient Platform for City-scale Smart Systems

Subhav Pradhan*, Abhishek Dubey*, Shweta Khare*, Fangzhou Sun*, Janos Sallai*,
Aniruddha Gokhale, Douglas Schmidt*, Martin Lehofer†, Monika Sturm‡
* Institute for Software Integrated Systems, Vanderbilt University, Nashville, TN, USA
† Siemens Corporate Technology, Princeton, NJ, USA
‡ Siemens AG Österreich, Vienna, Austria

I. Introduction

The advent of the Internet of Things (IoT) is driving several technological trends. The first trend is an increased level of integration between edge devices and commodity computers. This trend, in conjunction with low power-devices, energy harvesting, and improved battery technology, is enabling the next generation of information technology (IT) innovation: city-scale smart systems. These types of IoT systems can operate at multiple time-scales, ranging from closed-loop control requiring strict real-time decision and actuation to near real-time operation with humans-in-the-loop, as well as to long-term analysis, planning, and decision-making.

The second trend is an increased emphasis on *resilient deployment platforms*, which can be used to disperse computations closer to the physical phenomena. Two categories of these platforms can be distinguished based on the computing and communication resources they provide: (a) *Edge platforms* are low-power computing devices that are often available close to sensors and actuators, and (b) *Cloudlet platforms* are shared data centers that are managed locally, but can only provide a limited computing capacity. When these platforms are interconnected with a cloud or enterprise large-scale shared data center, they provide a unique opportunity for developing and deploying smart services [1].

While these platforms provide a key enabler for city-scale services, developing and operating these services becomes difficult due to the challenges associated with the remote, dynamic and distributed nature of these resources. This is especially difficult, considering that all critical services should remain available, even in vulnerable modes of operation, such as during faults, environmental changes, software updates, or cyber-attacks.

II. CHARIOT

For an IoT system to adapt to environmental changes or input stimuli, it needs a deployment platform that enables the analysis and management of (1) the constraints governing the system, *e.g.*, resource requirements or configuration restrictions, (2) the service requirements and provisioned resources that describe which services or APIs the software components either require from or provide to other components, and (3) the overall system goals describing the high-level services the system must provide. To meet these needs, our prior work [2], [3] presented CHARIOT, which is a deployment platform that encodes the configuration restrictions, service requirements, resource requirements, and goals in an IoT system as mathematical *constraints* on the system, its components, and possible (re)configurations.

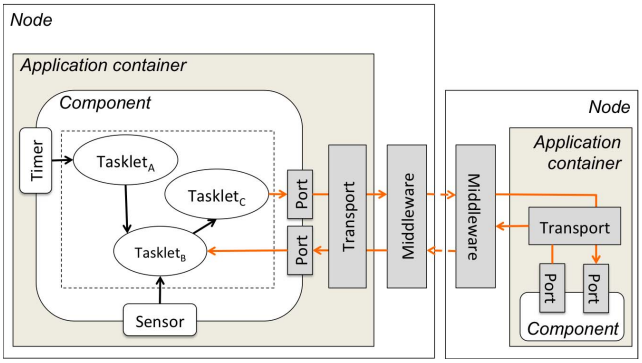

Fig. 1: Overview of the CHARIOT Component Model.

Our current work extends CHARIOT by reconceptualizing its distributed application management and resilience infrastructure as a two-layer hierarchy. The core idea behind this reconceptualization is establishing *resilience zones*, where each zone is characterized by the locality of the computing resources it comprises. CHARIOT initially limits reconfiguration actions to a specific zone and performs these reconfigurations via resilience engines specific to each zone. If no local reconfiguration is feasible, CHARIOT then propagates the reconfiguration to a higher-level resilience coordinator. This coordinator maintains a global view of the different resilience zones that it uses to govern inter-zonal reconfiguration actions.

A. Application Model

Apps supported by CHARIOT are component-based [4]. A component is a basic unit of functionality that can be composed and reused as part of different apps. Components communicate with other components in an app through well-defined interfaces that support common interaction patterns, such as the Request/Response pattern and the Publisher/Subscriber pattern. Interactions in these patterns can be synchronous or asynchronous. An app can be distributed across multiple physical nodes, in which case the components comprising that app will also be distributed across different

978-1-5090-3322-5/16 $31.00 © 2016 IEEE
DOI 10.1109/SEC.2016.28

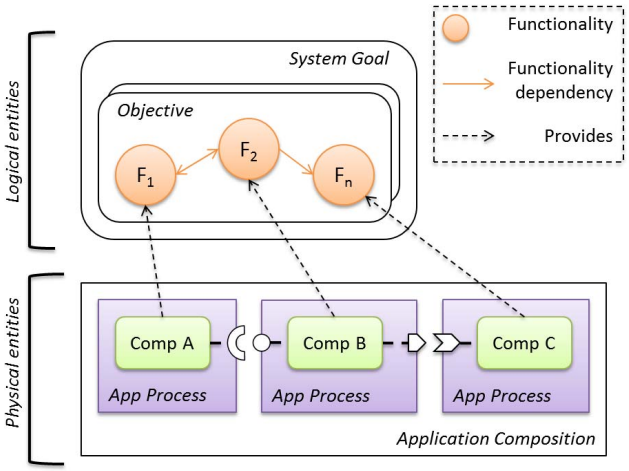

Fig. 2: Example of a Goal-based Description of an App.

physical nodes. Component logic is written as a data graph using *tasklets*, which are the smallest unit of computation and is equivalent to the concept of computation node in dataflow engines, such as Storm [5], Spark [6], TensorFlow [7], and Cloud Dataflow [8], see figure 1. Transport Objects decouple computation logic from the communication middleware logic.

B. Goal-Based System Description Model

CHARIOT uses the concept of goal-based system description to describe smart city systems. This abstraction is crucial since CHARIOT uses it to determine what goals must be met at any given time to consider one or more IoT systems to be active on a computing group collection. We therefore define *system goals* as high level functions that can be decomposed into smaller sub-functions (see Figure 2) using the concept of function decomposition [9]. A component provides a single leaf-level function and multiple components can implement the same functionality, thereby providing CHARIOT with more degrees of freedom to perform dynamic reconfiguration in response to failures or cyber-attacks.

C. Resilience Zone

Resilience is a system-level property - by definition any part of the system can fail, yet the system must be resilient and recover from failures. Hence, we need to work towards a solution that makes a system resilient, not just one more of its components. Therefore, at runtime the resilience engine is responsible for evaluating the current status of the goals of the system, the available resources and making necessary modifications to ensure the viability of the objectives. Our approach depends upon representing the current deployed configuration and the design space of possible alternatives as a constraint logic programming problem (CLP).

The encoding represents the problem as a set of constraints over integer variables, where a valuation of the variables represents a particular configuration of the system. Configuration choices and requirements, like 'component X can be mapped to nodes of type A and B' are also represented as Boolean expressions. The constraint-based representation can encode a potentially very large configuration space, as it does not encode configurations individually; rather it encodes them in an implicit, symbolic form. System goals are also represented by variables and their mapping to software components is represented via relations. The CLP representation also allows us to model the effect of faults. If a hardware node fails, a new constraint is added, representing the fact that no software component can be allocated to that node. Re-running the solver, a new configuration will be computed, that bypasses the failed component.

D. Resilience across multiple zones

In practice, a large scale city environment is not a single monolithic computation resource collection. Rather, it is a collection of resources distributed across multiple domains. Within the domains, the resource collections themselves can be of different categories. For example, they can be either low-powered edge devices, or cloudlets or clouds. A holistic management system will require to coordinate application maintenance across different resource groups. For this purpose, we are extending the CHARIOT management mechanism to support a two-level hierarchical resilience infrastructure.

The first level management mechanism is provided by a single resilience as described in previous section. The second level mechanism is provided by a *resilience coordinator*. The resilience coordinator is invoked by a resilience engine that cannot obtain a feasible answer to its local CLP. The resilience coordinator maintains information about different systems, their goals, associated objectives, and available resources. It solves a global CLP that can identify spare resources in another resilience zone and then moves the necessary objectives from one resilience zone to the other. If no migration is possible, it marks the appropriate zone as failed.

Acknowledgments: This work is sponsored in part by a research grant from Siemens Corporate Technology.

REFERENCES

[1] D. C. Schmidt, J. White, and C. D. Gill, "Elastic infrastructure to support computing clouds for large-scale cyber-physical systems," in *Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC), 2014 IEEE 17th International Symposium on*. IEEE, 2014, pp. 56–63.
[2] S. Pradhan, A. Dubey, and A. Gokhale, "Wip abstract: Platform for designing and managing resilient and extensible cps," in *2016 ACM/IEEE 7th International Conference on Cyber-Physical Systems (ICCPS)*. IEEE, 2016, pp. 1–1.
[3] S. M. Pradhan, A. Dubey, A. Gokhale, and M. Lehofer, "Chariot: A domain specific language for extensible cyber-physical systems," in *Proceedings of the Workshop on Domain-Specific Modeling*. ACM, 2015, pp. 9–16.
[4] G. T. Heineman and B. T. Councill, *Component-Based Software Engineering: Putting the Pieces Together*. Reading, Massachusetts: Addison-Wesley, 2001.
[5] Apache Software Foundation, "Apache Storm," http://storm.apache.org/.
[6] ——, "Apache Spark," http://spark.apache.org/.
[7] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous systems, 2015," *Software available from tensorflow. org*.
[8] Google, "Cloud Dataflow," https://cloud.google.com/dataflow/.
[9] T. Kurtoglu, I. Y. Tumer, and D. C. Jensen, "A functional failure reasoning methodology for evaluation of conceptual system architectures," *Research in Engineering Design*, vol. 21, no. 4, pp. 209–234, 2010.