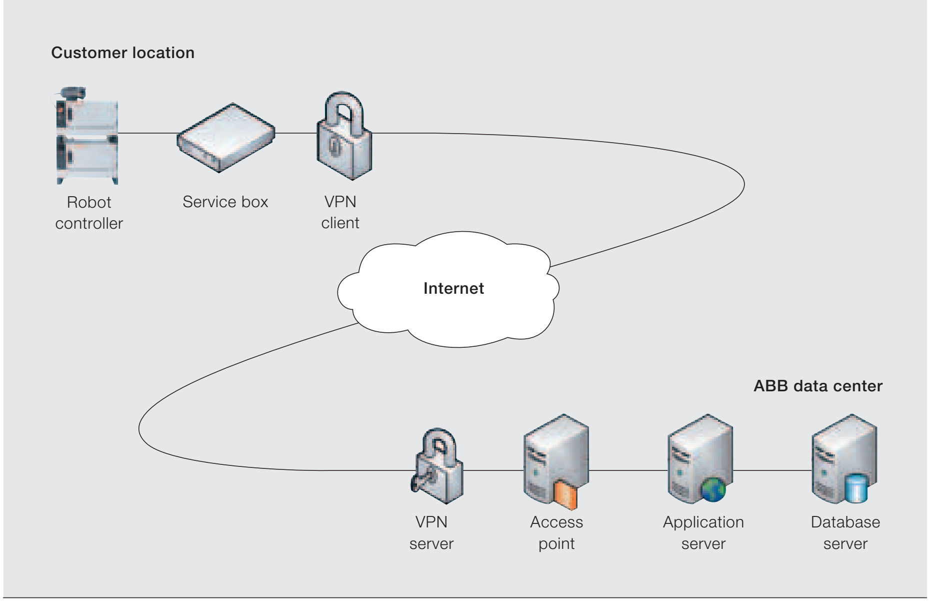
Scaling factors

Software scalability for ABB's future IT

THIJMEN DE GOOIJER, ANTON JANSEN, HEIKO KOZIOLEK, STEVE MURPHY – Robots are hard-working and valuable members of any production line and any time they are out "sick" can have a significant negative effect on productivity. For this reason, ABB offers robot service agreements that provide assistance to customers in case of failure as well as preventive maintenance to reduce the frequency of failures. This service has been enhanced by letting the robots communicate with the customer and to ABB so that timely interventions can be made. Since 2007, when this feature was introduced, more and more robots have started communicating. Not only that, but the number of related services offered has increased, too. These factors have increased the load on the infrastructure, so ways to optimize the back-end scalability and performance requirements have been studied.

1 High-level software architecture of the current remote service platform



It would be very troublesome if one were to build a pyramid and, upon completion, discover it should now be made 10 percent larger – unless the construction plan had been designed with scalability in mind. Similar scalability challenges are faced in industry and a good case in point is the ABB back-end system that handles remote diagnostic communication with robots under service agreement.

This back-end system, known as the robotics remote diagnostic platform, consists for four tiers, the first of which is the service box connected to the robot controller itself → 1. This box communicates with the other tiers, the so-called back-end, which are situated at ABB. This back-end consists of an access point, an application server and a database. The access point secures the data flow between the robot service box at the customer site, and the application server and database on ABB's internal network. The application server, the heart of the remote service system, performs the necessary diagnostics. All data is stored in the database.

Increasing demand from robots in the field, coupled with elevated levels of service and performance, means that ever more service boxes interact with these three back-end elements. This continually increasing traffic makes it all the more important that a defined methodology for scaling up the back-end capacity is in place. A capacity increase in the back-end requires careful redesign of the three tiers, taking into account the interaction between them. Ideally, the capacity of all three back-end tiers would be identical; if the capacity of one tier is too low, a bottleneck will arise, slowing the other tiers and resulting in generally slower response times and delays.

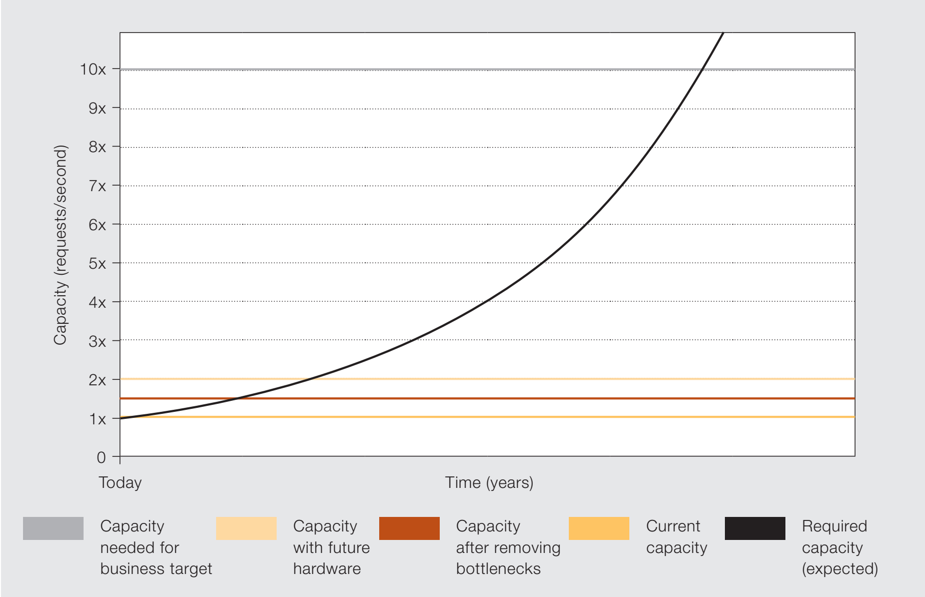
Increasing demand from robots in the field, coupled with elevated levels of service and performance, means ever more interaction.

When work to examine scalability methodology began, the remote diagnostic platform was already heavily loaded, with capacity just slightly higher than demand. Initial investigations quickly identified and eliminated bottlenecks, increasing capacity by 56 percent → 2. Hardware upgrades were planned to further boost capacity, but any gains from these would be eroded in one to three years by increased demand and further extensions to the services offered. Actually, to support the long-term business target, a tenfold capacity increase is needed. This calls for careful engineering of the software architecture and a roadmap to guide the system's transformation.

Three steps were taken to establish a roadmap for system scalability. First, the performance of the current version was measured to identify bottlenecks. Software bottlenecks occur when one software routine, for example, a Web service, is stopped or slowed down because it has to wait on another, slower routine. Software bottlenecks inhibit efficient hardware use. Second, a performance model of the system architecture and a cost model for the system resources were created. Third, a performance simulation was used to evaluate how different design alternatives perform under the predicted number of remotely connected robots. Based on this model-based stress testing, a roadmap comprising multiple cost-efficient steps toward improved capacity and scalability was produced.

Title picture
IT infrastructure has to be scalable to efficiently accommodate the rising number of robots "reporting in."

2 Capacity requirements of the robotics remote service platform



To support the long-term business target, a tenfold capacity increase is needed.

Bottlenecks

Before an accurate performance model can be created, the system performance itself must be understood and any bottlenecks have to be identified. This was done using a dynaTrace[1] application that features integrated application performance management (APM) and performance profiling. This tool was selected from a list of 58 tools for its good .NET platform support, automatic source code instrumentation and its distributed transaction tracing technology ➔ 3. Server log analysis was used to discover which transactions are performance-critical and occur most. This information was input into a load generator application to simulate a realistic workload on the system in the experimental environment. During these experiments various performance measurements (eg, CPU load, network delays) were obtained.

Planning for future capacity needs is not just a matter of performance – there are hardware, software, and hosting costs too.

Performance model

To model performance, data were fed into a software architecture simulation tool called Palladio[2]. Palladio's modeling language uses a notation similar to the unified modeling language (UML), a de facto industry standard, which makes it easy to create, use and communicate model and simulation results[3]. Models can be reused and rearranged to investigate alternative software architectures. While constructing the performance model, trade-offs between abstraction, detail, and prediction accuracy were continuously made. The first model recreated the current system and this was calibrated so that the model predictions matched the measured performance. In later stages, the robustness of the model was increased to ensure accuracy of the predictions for varying workloads and different system designs.

Cost model

One obvious way to avoid the risk of resource under-provision, and consequent customer dissatisfaction, is to massively over-provide. However, planning for future capacity needs is not just a matter of performance as hardware, software and hosting costs also need to be taken into account; idle resources cost money. To capture this second dimension, a cost model, based on several price lists, was created. This model specifies the cost of the resources in the performance model.

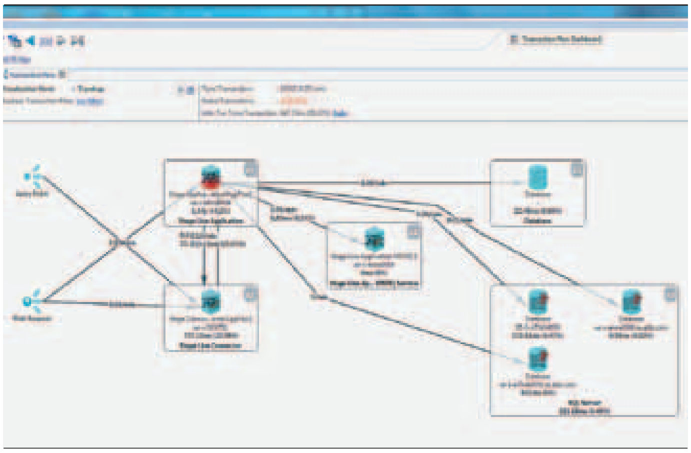
Footnotes

[1] http://www.dynatrace.com/

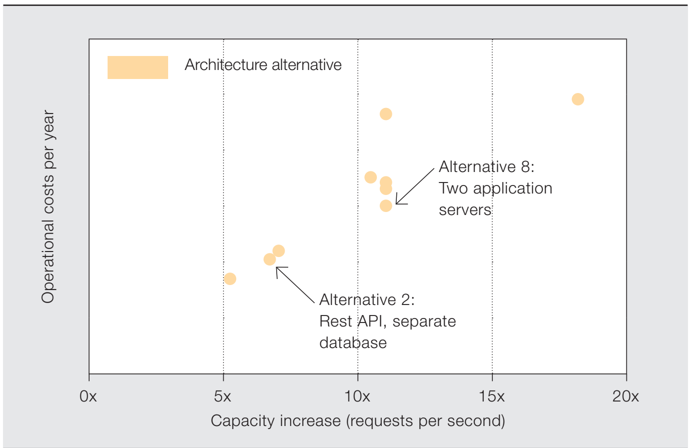
[2] http://www.palladio-simulator.com/inventory

[3] http://en.wikipedia.org/wiki/Unified_Modeling_Language

3 A dynaTrace screenshot showing response time monitoring and request flows between servers



4 Costs of overcapacity per evaluated architecture candidates



If the performance model alone were used to construct the roadmap, it could advise only on actions to increase capacity. By combining it with the cost model, the costs can also be taken into account, creating a cost-efficient transition schedule that guarantees timely action to satisfy capacity demands.

Architecture alternatives and migration roadmap

Over a dozen architectures that accommodated the desired tenfold capacity increase were considered. Approaches included, for example, cloning specific services; splitting the responsibilities of existing services so that they can be better distributed between multiple servers; and separating incoming requests into different classes that can be processed by dedicated servers. Actually implementing and deploying the alternatives suggested by the modeling in order to measure their actual performance would be too expensive. However, it was possible to model each of them by altering the performance model. Each alternative was simulated and its capacity in terms of supported requests per second determined. Similarly, the cost model delivered the expected annual operational costs for each alternative ➔ 4.

Alternatives 2 and 8 are highlighted in the figure as they are the most cost-effective solutions for providing a fivefold and tenfold capacity increase, respectively. Alternative 2 suggests implementing a REST API (a type of distributed Web service) for the system and deploying the database onto a dedicated server. Alternative 8 suggests replicating the application server, which hosts several services, and using a load balancing mechanism to distribute the incoming requests evenly.

Directly migrating to alternative 8 would be wasteful because the capacity it supports is not needed for several years. In addition, alternatives 2 and 8 are compatible with each other, ie, they may be combined to reach a higher capacity. Thus, the current plan is to migrate the system to alternative 2 and then, later, to alternative 8. This provides the most cost-effective way to fulfill the target requirement of a tenfold capacity increase.

Increasing need for IT capacity planning

The approach to scalability presented here is very suitable for the growing demands arising from both the increased levels of service being offered and the increasing number of robots talking to the ABB back-end system. It can also cater to the demand arising from the ease of retrofitting remote diagnostics to an installed base. The scalable software architecture discussed can serve as a blueprint for other ABB remote diagnostics offerings where more powerful hardware is simply not sufficient to reach the target capacity, as well as for future remote service solutions.

With the advent of the Internet of things (IoT)[4], the relevance of highly-scalable back-end systems is increasing for ABB. Scalable systems will have to be created at multiple levels in the IoT to keep performance for end-users at the expected level and to make sure that customers with smart homes and factories will not experience delays in remote communications. The experience gained in this engineering approach to scalability will help ABB build scalable systems cost-effectively and to tackle performance problems as they arise.

Thijmen de Gooijer
Heiko Koziolek
ABB Corporate Research
Ladenburg, Germany
thijmen.de-gooijer@de.abb.com

Anton Jansen
ABB Corporate Research
Västerås, Sweden
anton.jansen@se.abb.com

Steve Murphy
ABB Robotics
Mölndal, Sweden
steve.murphy@se.abb.com

Footnote

[4] http://en.wikipedia.org/wiki/Internet_of_Things