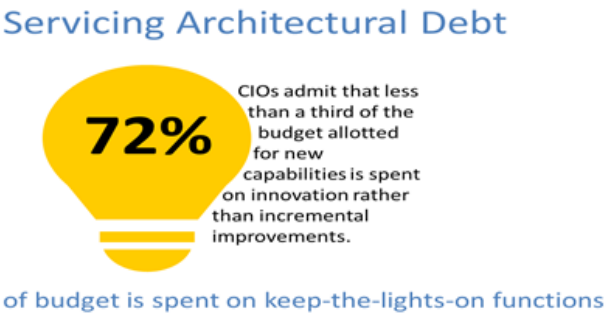
If apps incur technical debt then networks incur architectural debt

Lori MacVittie, 2014-23-06

#devops #sdn #SDDC #cloud

72%. That's an estimate of how much of the IT budget is allocated to simply keeping the lights on (a euphemism for everything from actually keeping the lights on to cooling, heating, power, maintenance, upgrades, and day to day operations) in the data center.



> In a recent Forrester Research survey of IT leaders at more than 3,700 companies, respondents estimated that they spend an average 72% of the money in their budgets on such keep-the-lights-on functions as replacing or expanding capacity and supporting ongoing operations and maintenance, while only 28% of the money goes toward new projects.
>
> How to Balance Maintenance and IT Innovation

This number will not, unfortunately, significantly improve without intentionally attacking it at its root cause: architectural debt.

Data Center Debt

The concept of "debt' is not a foreign one; we've all incurred debt in the form of credit cards, car loans and mortgages. In the data center, this concept is applied in much the same way as our more personal debt - as the need to "service" the debt over time.

Experts on the topic of technical debt point out that this "debt' is chiefly a metaphor for the long-term repercussions arising from choices made in application architecture and design early on.

> **Technical debt** is a neologistic metaphor referring to the eventual consequences of poor software architecture and software development within a codebase. The debt can be thought of as work that needs to be done before a particular job can be considered complete. If the debt is not repaid, then it will keep on accumulating interest, making it hard to implement changes later on. Unaddressed technical debt increases software entropy.
>
> Wikipedia

This conceptual debt also occurs in other areas of IT, particularly those in the infrastructure and networking groups, where architectural decisions have long lasting repercussions in the form of not only the cost to perform day-to-day operations but in the impact to future choices and operational concerns. The choice of a specific point product today to solve a particular pain point, for example, has an impact on future product choices. The more we move toward software-defined architectures - heavily reliant on integration to achieve efficiencies through automation and orchestration - the more interdependencies we build. Those interdependencies cause considerable complexity in the face of changes that must be made to support such a loosely coupled but highly integrated data center architecture.

We aren't just maintaining configuration files and cables anymore, we're maintaining the equivalent of *code* - the scripts and methods used to integrated, automate and orchestrate the network infrastructure.

Steve McConnell has a lengthy blog entry examining technical debt. The perils of not acknowledging your debt are clear:

> *One of the important implications of technical debt is that it must be serviced, i.e., once you incur a debt there will be interest charges. If the debt grows large enough, eventually the company will spend more on servicing its debt than it invests in increasing the value of its other assets.*

Debt must be serviced, which is why the average organization dedicates so much of its budget to simply "keeping the lights on." It's servicing the architectural debt incurred by a generation of architectural decisions.

Refinancing Your Architectural Debt

In order to shift more of the budget toward the innovation necessary to realize the more agile and dynamic architectures required to support more things and the applications that go with them, organizations need to start considering how to shed its architectural debt.

First and foremost, software-defined architectures like cloud, SDDC and SDN, enable organizations to pay down their debt by automating a variety of day-to-day operations as well as traditionally manual and lengthy provisioning processes. But it would behoove organizations to pay careful attention to the choices made in this process, lest architectural debt shift to the technical debt associated with programmatic assets. Scripts are, after all, a simple form of an application, and thus bring with it all the benefits and burdens of an application.

For example, the choice between a feature-driven and an application-driven orchestration can be critical to the long-term costs associated with that choice. Feature-driven orchestration necessarily requires more steps and results in more tightly coupled systems than an application-driven approach. Loose coupling ensures easier future transitions and reduces the impact of interdependencies on the complexity of the overall architecture. This is because feature-driven orchestration (integration, really) is highly dependent on specific sets of API calls to achieve provisioning. Even minor changes in those APIs can be problematic in the future and cause compatibility issues. Application-driven orchestration, on the other hand, presents a simpler, flexible interface between provisioning systems and solution. Implementation through features can change from version to version without impacting that interface, because the interface is decoupled from the actual API calls required.

Your choice of scripting languages, too, can have much more of an impact than you might think. Consider that a significant contributor to operational inefficiencies today stems from the reality that organizations have an L4-7 infrastructure comprised of not just multiple vendors, but a wide variety of domain specificity. That means a very disparate set of object models and interfaces through which such services are provisioned and configured. When automating such processes, it is important to standardize on a minimum set of environments. Using bash, python, PERL and juju, for example, simply adds complexity and begins to fall under the Law of Software Entropy as described by Ivar Jacobson et al. in "Object-Oriented Software Engineering: A Use Case Driven Approach":

> The second law of thermodynamics, in principle, states that a closed system's disorder cannot be reduced, it can only remain unchanged or increased. A measure of this disorder is entropy. This law also seems plausible for software systems; as a system is modified, its disorder, or entropy, always increases. This is known as **software entropy**.

Entropy is the antithesis of what we're trying to achieve with automation and orchestration, namely the acceleration of application deployment. Entropy impedes this goal, and causes the introduction of yet another set of systems requiring day-to-day operational attention.

Other considerations include deciding which virtual overlay network will be your data center standard, as well as the choice of cloud management platform for data center orchestration. While such decisions seem, on the surface, to be innocuous, they are in fact significant contributors to the architectural debt associated with the data center architecture.

Shifting to Innovation

Every decision brings with it debt; that cannot be avoided. The trick is to reduce the interest payments, if you will, on that debt as a means to reduce its impact on the overall IT budget and enable a shift to funding innovation.

Software-defined architectures are, in a way, the opportunity for organizations to re-finance their architectural debt. They cannot forgive the debt (unless you rip and replace) but these architectures and methodologies like devops can assist in reducing the operational expenses the organization is obliged to pay on a day-to-day basis.

But it's necessary to recognize, up front, that the architectural choices you make today do, in fact, have a significant impact on the business' ability to take advantage of the emerging app economy. Consider carefully the options and weigh the costs - including the need to service the debt incurred by those options - before committing to a given solution.

Your data center credit score will thank you for it.

F5 Networks, Inc. | 401 Elliot Avenue West, Seattle, WA 98119 | 888-882-4447 | f5.com

F5 Networks, Inc.	F5 Networks	F5 Networks Ltd.	F5 Networks
Corporate Headquarters	Asia-Pacific	Europe/Middle-East/Africa	Japan K.K.
info@f5.com	apacinfo@f5.com	emeainfo@f5.com	f5j-info@f5.com