# Using Automated Planning in data centers fault tolerance systems

**Douglas Trajano**
Master's Degree in Computer Science
Pontifical Catholic University of Rio Grande do Sul - PUCRS
School of Technology. Porto Alegre, Brazil
douglas.trajano@edu.pucrs.br

**Abstract**

Data centers are key components for several companies and the digital transformation increases the quantity and the complexity of these IT assets. Develop intelligent systems that can be used in hardware failures is an area of study of artificial intelligence. With it in mind, we propose an automated planner to act or guide engineers in critical incidents to keep data centers healthy.

## 1 Introduction

Nowadays, IT operations are crucial for business continuity. Information systems are used by companies in different processes and products, all these services are implemented in data centers. Critical incidents in data centers may impact the usability of these services and it needs to be handled in timely fashion by the engineering team to avoid financial losses or affect their end-users.

A data center is a building composed of computer systems, storage systems, infrastructure for power supply, telecommunications, environmental controls (e.g., air conditioning, fire suppression), and many other associated components. The complexity of these resources lies in the fact that all these components need to work together, if one of these components has an issue, it can affect other components. Engineering teams need to have a good knowledge of these components and their specific characteristics to be able to solve incidents. Knowledge bases for known issues are built to help engineers to apply solutions correctly and as soon as possible.

In this work, we propose an automated planner that can understand known issues and provide a set of ordered actions to be implemented in the architecture to fix issues or apply workarounds solutions. We use Planning Domain Definition Language (PDDL) to model this domain and some problems (known issues). For each problem we can have different goals depends on the criticality of the situation. Nonetheless, our automated planner will provide a plan with the actions that are expected to achieve the goal.

This work is organized as follows. Section 2 explains our approach in this work. Section 3 provides a detailed description of the proposed solution. Section 4 explains the experimental process and the results in detail. Section 5 discusses related works. Finally, section 6 presents our conclusions and possibilities for future work.

Copyright © 2021, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

## 2 Approach

AI Planning is a field of Artificial Intelligence that explores the process of using autonomous techniques to solve planning and scheduling problems. A planning problem is one in which we have some initial starting state, which we wish to transform into the desired goal state through the application of a set of actions.(Green 2021)

PDDL (Planning Domain Definition Language) is intended to express the "physics" of a domain, that is, what predicates there are, what actions are possible, what the structure of compound actions is, and what the effects of actions are. (Ghallab et al. 1998)

We use PDDL to modeling our domain and some sample problems (known issues). The domain contains some components like worker computers, master computers, switches, storage devices, etc. interpreted as objects. The relationship between the components and their states is defined as predicate properties (properties that are either true or false). All possible actions that our planner can apply are defined as well in the domain file. Each problem definition contains the instances of our objects, the initial states and the goals for that problem.

## 3 Implementation

The domain and the problems are defined using PDDL. The automated planner and validator used in this project are provided by planning.domains[1]. The source code created of this project are available in GitHub [2].

As explained early, data centers are complex systems with several hardware components, to validate our proposed solution, we focused on modeling the main components and their integration.

Our domain is composed of components of three types. Each component has predicates and actions.

[1] http://solver.planning.domains/

[2] https://github.com/DougTrajano/pucrs-ap-data-center

### 3.1 Computer components

The computer resources defined in our domain are worker and master nodes.

Worker computers are used to deploy applications and are controlled by master computers through switches.

**Predicates**

- **worker-healthy**: Indicates if the worker is healthy or not.
- **worker-high-mem**: Indicates a high usage of memory.
- **worker-high-cpu**: Indicates a high usage of processor.
- **worker-high-network**: Indicates a high usage of network.

**Actions**

- **worker-turn-on**: Turn on the worker node.
- **worker-turn-off**: Turn off the worker node.

Master computers provide management tools for an entire cluster. All worker nodes attached in a switch will use these tools.

**Predicates**

- **master-healthy**: Indicates if the master is healthy or not.

**Actions**

- **master-turn-on**: Turn on the master node.
- **master-turn-off**: Turn off the master node.

### 3.2 Network components

Routers and switches are networks devices used in our implementation. All others components are connected by switches and expose to the external world using the routers.

Routers expose clusters to the outside world, they work in pairs.

**Predicates**

- **router-healthy**: Indicates if the router is healthy or not.
- **router-pair**: Defines an integration between two routers.

**Actions**

- **router-turn-on**: Turn on the router.
- **router-turn-off**: Turn off the router.
- **router-mesh-on**: Define a mesh with two routers.
- **router-mesh-off**: Unmesh the routers.

Switches connect all components of the cluster and is attached in a router.

**Predicates**

- **switch-healthy**: Indicates if the switch is healthy or not.
- **switch-attach-router**: Establish a connection between a switch and a router.
- **switch-attach-master**: Establish a connection between a switch and a master computer.
- **switch-attach-worker**: Establish a connection between a switch and a worker computer.

**Actions**

- **switch-turn-on**: Turn on the switch.
- **switch-turn-off**: Turn off the switch.

### 3.3 Storage components

We have two types of storage devices.

EBS is a block store used by computers for their operating systems. Each EBS can be used by only one computer.

**Predicates**

- **ebs-healthy**: Indicates if the EBS is healthy or not.
- **ebs-locked**: Indicates if the EBS is locked or not.
- **ebs-attached**: Establish a connection between an EBS and a computer.

**Actions**

- **ebs-turn-on**: Turn on the EBS.
- **ebs-turn-off**: Turn off the EBS.

The Bucket represents object storage attached to a master computer and shared with applications deployed in that cluster.

**Predicates**

- **bucket-healthy**: Indicates if the bucket is healthy or not.
- **bucket-locked**: Indicates if the bucket is locked or not.
- **bucket-attached**: Establish a connection between a bucket and a master computer.

**Actions**

- **bucket-turn-on**: Turn on the bucket.
- **bucket-turn-off**: Turn off the bucket.

The diagram below demonstrates how these components work together in a healthy scenario.

Each cluster receives the name of a famous scientist as a tribute. This has no practical effect on our architecture.

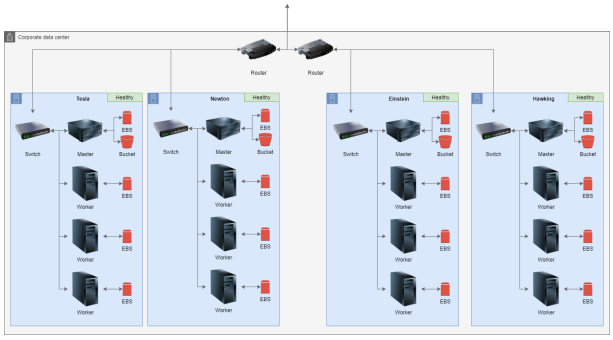


Figure 1: Data Center Domain

## 4 Experiments

We want to validate that PDDL has the flexibility to modeling different architectures of data centers, with more or fewer components, and different known issues that our engineers can define and maintain it using a common language. So, engineering teams can act more quickly on incidents using the planner as a guide or allowing that the planner can apply the actions directly in the architecture.

Three different scenarios are explored in this work. As PDDL uses the domain and problem as definitions, we will reference these scenarios as problems.

The goal of our three problems are the same, all components healthy and connected properly and no error message.

## 4.1 Problem 1

The first problem explore the initial state of our data center. All components are turned off and need to be turned on in a specific order. Firstly, routers and switches need to be turned on, they are responsible to communicate with others objects. Worker and master nodes require that storage devices are turned on previously.

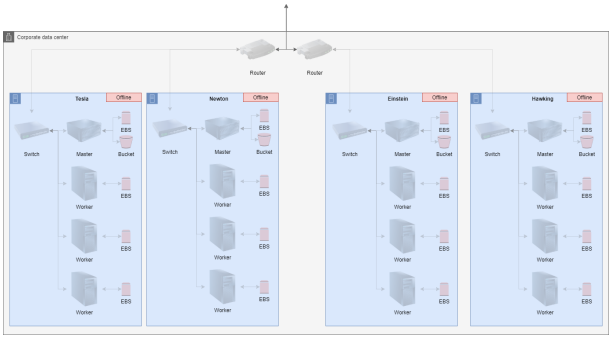

Figure 2: Problem 1

The plan for this problem is shown below. Note that the solver respects the dependencies between the components, starting routers and switches firstly and the storage and computers in the sequence.

**router-turn-on** router1
**switch-turn-on** switch1 router1
**switch-turn-on** switch2 router1
**switch-turn-on** switch3 router1
**switch-turn-on** switch4 router1
**router-turn-on** router2
**router-mesh-on** router1 router2
**ebs-turn-on** ebs1
**worker-turn-on** w12 switch4 ebs1
**ebs-turn-on** ebs2
**worker-turn-on** w11 switch4 ebs2
**ebs-turn-on** ebs3
**worker-turn-on** w10 switch4 ebs3
**ebs-turn-on** ebs4
**worker-turn-on** w9 switch3 ebs4
**ebs-turn-on** ebs5
**worker-turn-on** w8 switch3 ebs5
**ebs-turn-on** ebs6
**worker-turn-on** w7 switch3 ebs6
**ebs-turn-on** ebs7
**worker-turn-on** w6 switch2 ebs7
**ebs-turn-on** ebs8
**worker-turn-on** w5 switch2 ebs8
**ebs-turn-on** ebs9
**worker-turn-on** w4 switch2 ebs9
**ebs-turn-on** ebs10
**worker-turn-on** w3 switch1 ebs10
**ebs-turn-on** ebs11
**worker-turn-on** w2 switch1 ebs11
**ebs-turn-on** ebs12
**worker-turn-on** w1 switch1 ebs12
**ebs-turn-on** ebs13
**ebs-turn-on** ebs14
**ebs-turn-on** ebs15
**ebs-turn-on** ebs16
**bucket-turn-on** bucket1
**master-turn-on** hawking switch4 ebs13 bucket1
**bucket-turn-on** bucket2
**master-turn-on** einstein switch3 ebs14 bucket2
**bucket-turn-on** bucket3
**master-turn-on** newton switch2 ebs15 bucket3
**bucket-turn-on** bucket4
**master-turn-on** tesla switch1 ebs16 bucket4

## 4.2 Problem 2

In the second problem, we have some components in three different clusters with errors (memory, CPU and network). When the computer is restarted, its solved this issue.

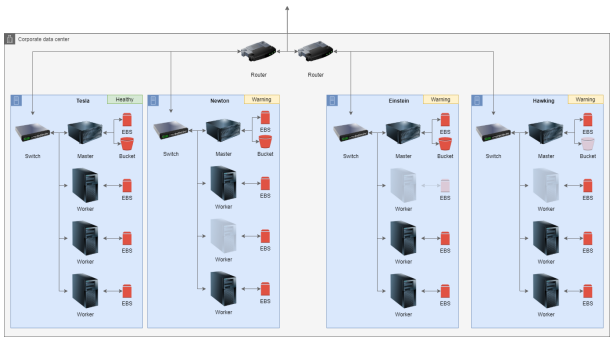

Figure 3: Problem 2

Some computers that are facing issues are healthy, but with error messages triggered. We defined only actions for turn on and turn off the components, so the solution found by the planner is to turn off the problematic computers and turn on them again. The plan for this problem is shown below.

**worker-turn-on** w7 switch3 ebs10
**bucket-turn-on** bucket4
**worker-turn-off** w5 switch2 ebs7
**worker-turn-on** w5 switch2 ebs7
**master-turn-off** hawking switch4 ebs13 bucket4
**master-turn-on** hawking switch4 ebs13 bucket4
**worker-turn-off** w10 switch4 ebs14
**worker-turn-on** w10 switch4 ebs14

## 4.3 Problem 3

In the third problem, we have a cluster complete offline, all components need to be turned on.

The planner provided a sequence to turn on all the components related to the tesla cluster. The plan for this problem is shown below.

**switch-turn-on** switch1 router1

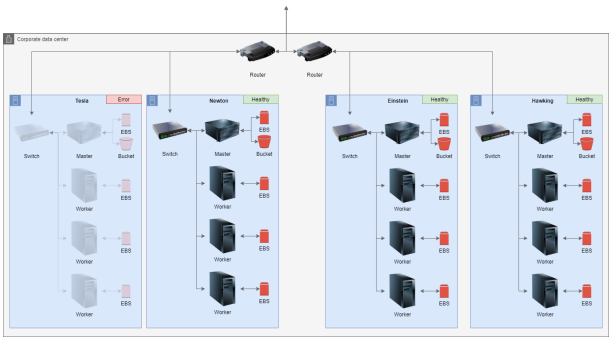


Figure 4: Problem 3

**ebs-turn-on** ebs1
**worker-turn-on** w3 switch1 ebs1
**ebs-turn-on** ebs2
**worker-turn-on** w2 switch1 ebs2
**ebs-turn-on** ebs3
**worker-turn-on** w1 switch1 ebs3
**ebs-turn-on** ebs4
**bucket-turn-on** bucket1
**master-turn-on** tesla switch1 ebs4 bucket1

## 5 Related work

Few works addressed the usability of artificial intelligence systems in fault tolerance.

In "Predicting Remediations for Hardware Failures in Large-Scale Datacenters" (Lin et al. 2020) a Gradient Boosted Decision Tree is trained using as input data server failure logs, tooling logs, and server attributes for all solved incidents.

In "Predicting Computer System Failures Using Support Vector Machines" (Fulp, Fink, and Haack 2008) the authors describe the usage Support Vector Machine (SVM) approach to predict failure events based on system log files.

These works are great and help to improve data center management. Our work is distinguished by providing a definition language that engineers can add new known issues very quickly to the system. Also, the explainability of the automated planner is a great benefit as it can be easily interpreted.

## 6 Conclusions

The idea of using PDDL to formalize domains of data center architectures seems to be promising. Nonetheless, there are still many questions to be answered.

- Is it reasonable to require PDDL skills for engineers? Can engineers develop and maintain the domain and problem files?
- How the information represented by predicates will be collected for the planner?
- Can we allow the planner to act directly on the infrastructure? Should some actions be restricted to engineers?

As the future work, is interesting to explore more complex architectures, with more components, involving applications and more possible situations (problems). Define more appropriate actions for the components will provide better plans avoiding a sequence of the turn off and turn on that can be stressful for the system. This work is just a small step towards a new possibility to manage the infrastructure of our data centers.

## References

Fulp, E. W.; Fink, G. A.; and Haack, J. N. 2008. Predicting computer system failures using support vector machines. In *Proceedings of the First USENIX Conference on Analysis of System Logs*, WASL'08, 5. USA: USENIX Association.

Ghallab, M.; Howe, A.; Knoblock, C.; Mcdermott, D.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL—The Planning Domain Definition Language.

Green, A. 2021. What is AI Planning? - Planning.wiki - The AI Planning & PDDL Wiki. (Accessed on 06/26/2021).

Lin, F.; Davoli, A.; Akbar, I.; Kalmanje, S.; Silva, L.; Stamford, J.; Golany, Y.; Piazza, J.; and Sankar, S. 2020. Predicting remediations for hardware failures in large-scale datacenters. In *2020 50th Annual IEEE-IFIP International Conference on Dependable Systems and Networks-Supplemental Volume (DSN-S)*, 13–16. IEEE.