Cloud Migration using Automated Planning

Maja Vukovic and Jinho Hwang
IBM T.J. Watson Research Center
Yorktown Heights, NY
Email: {maja, jinho}@us.ibm.com

***Abstract*—Cloud migration transforms company's data, applications and services to (or between) one or more other Cloud environments. Enterprises are increasingly migrating their IT infrastructures to Cloud, given the appeal of (pay-per-use) elastic resources. Yet, existing IT infrastructures are complex, heterogeneous and dynamic ecosystems. As a result, there is no single standardized process to seamlessly manage migration at enterprise scale, and often significant level of manual intervention is required, both in reasoning about migration and during its execution.**

This paper presents a system that automates the process of migration to Cloud. It embeds a Metric-FF Artificial Intelligence (AI) planning algorithm to dynamically assemble migration plans based on the properties of source and target environments, as well as available migration tooling. The paper describes the challenges in migration planning, AI domain design for migration. This work demonstrates that the system provides an effective and scalable solution to generating plans based on the source environment of 700 servers, and varying size of the migration service requests.

Keywords−Cloud Migration, IT Service Management, Automated Planning, Business Process Management

I. Introduction

Driven by the promise of low-cost access to elastic resources, enterprises are increasingly moving their existing IT infrastructure to Cloud [1]. Depending on the compliance and security requirements, as well as the business criticality of the applications (often referred to as workloads), enterprises may choose one or more Cloud environments, such as private cloud, 3rd party public cloud, traditional data center, etc., resulting in hybrid Cloud environments.

Hybrid Cloud environments are characterized by dynamicity and heterogeneity of resources. As a result, IT service management processes, such as IT transformation (i.e. migration of workloads and images), are increasing in complexity. These processes now need to seamlessly and securely operate across (multiple) hybrid Cloud environments; and allow for integration with multiple service providers selected by the customer. Despite the availability of the formalized guidelines, the traditional IT service management processes are still executed in ad-hoc and non-standardized manner. This results in a significant level of manual intervention and supervision, and impacts duration and quality of execution of the processes, also increasing dependency on the skill level.

Business process management (BPM) practices offer an approach to management of complexity in such dynamic and hybrid environments with systematic development processes [2]. At the same time, while Web-friendly technologies, such as REpresentational State Transfer (REST) greatly simplify service reusability, and through Application Programming Interfaces (APIs) increase consumability [3], the process automation has not been successfully implemented in the hybrid Cloud. However, this trend opens up an opportunity for process automation and orchestration in hybrid Cloud through API-enabled services. For example, Infrastructure-as-a-Service (IaaS) provides comprehensive REST APIs to manage the computing and network resources. Also, Platform-as-a-Service (PaaS) providers manage the IT process automation using APIs, while Software-as-a-Service (SaaS) providers supply specialized services through API-enabled processes.

In this paper, we present a service-based system, which employs BPM technology to orchestrate automated services via APIs and human-driven services, to provide large-scale parallelism of enterprise migration workflows. From a practical prospective, this BPM-enabled approach reduces the elapsed process time and handovers between teams through process orchestration. It also increases efficiency of the migration through automation of individual process steps. The system also embeds an Artificial Intelligence (AI) planner to dynamically generate migration workflows, which provides a mechanism to handle the increasing complexity of source and target environments during migration process.

This paper makes two contributions:

1) It describes a novel application of automated artificial intelligence (AI) planning to dynamically generate migration process plans, and handle exceptions during Cloud migration service execution.
2) It discusses challenges that need to be overcome to support dynamism and variability in planning of the migration to Cloud workflows.

The rest of the paper is structured as follows. Section II described the approach to migration to Cloud environments and challenges in the current practice, stemming from complexity of environment and dependence on human tasks. Section III presents an overall approach to automating migration to Cloud and our service-oriented system. Section IV describes how we employ goal-oriented planning technology to drive dynamic creation and adaptation of migration plans. Section V presents a set of experiments that evaluate feasibility of AI planning for migration service. It also discusses the key challenges in integrating and automating the entire migration process. Section VI puts our work in the context of related research in migration, business process management and automation of service management. Section VII concludes and sets out the direction for future work.

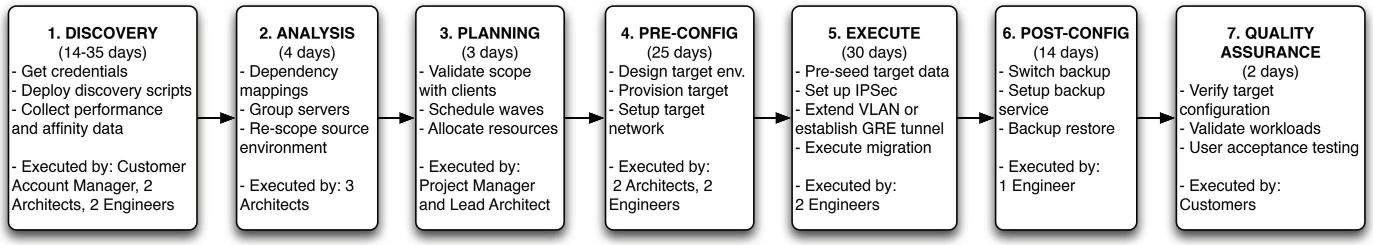


Fig. 1: High-level process overview for migration to Cloud

II. Process of Migration to Cloud

Migration to Cloud may transition company's data, applications and services from on-site premises to a cloud environment, or between one or more Clouds. Migration techniques are selected manually based on experiential knowledge of practitioners, and as a result migration process is highly error-prone.

Figure 1 depicts the typical migration process. It starts with the discovery stage, followed by the analysis of the uncovered source environment and evaluation of its fit to the target. For example, can the discovered resources be moved to the target Cloud as they are or do they require some level of adaptation? Planning stage groups similar servers or workloads into so-called "waves" that will be scheduled for migration. Before migration is executed, pre-configuration is performed, which includes provisioning of target environment and network setup. Once the migration is completed, post-configuration tasks are executed, such as backup switching. The process completes with the quality assurance step to ensure that the new environment is operational.

Technical challenges in migration and its automation [4] are driven by the heterogeneity of the source and target environments, number of available discovery, planning and migration tools and many unanticipated events and exceptions that can derail the process. For example, availability of a large variety of server platforms impacts the choice of migration techniques that need to be considered. On-premise servers likely run on different platforms, different physical boxes and various hypervisors that involve different image formats. Not only platforms, but also operating systems have different architectures. As a result, there is no one-size fits all migration plan or tool [5]. Researchers investigated how to plan for the migration of enterprise-scale on-premise data centers with large number of servers and complexity in server connectivity [6], and automatically discover server groups that represent business workloads [7].

At each step of migration process, practitioners have a choice of one or more tools. Image based migration, is a per-unit based method, such as physical to virtual conversion. Example of existing tools for this migration type includes VMware vConverter, VMware vReplicator, VMware Site Recovery Manager (SRM), and Rackware. Post-configuration may involve use of eVault or IBM Tivoli Storage Manager (TSM), further increasing the complexity and number of choices at each step of the migration process.

Furthermore, the migration process never executes seamlessly in a sequential manner, due to many unanticipated events at each step. For example, at the discovery stage, there may be an issue with the accuracy of the discovered data. The change in scope, at the analysis step may delay further execution. The pre-configuration may be delayed due to network circuits not being ready. During migration, the source environment may have changed or secure WAN may not be available. In summary, the IT environments are highly dynamic, and the initially discovered infrastructure may no longer be accurate at the time of migration execution.

III. Cloud Migration Orchestrator (CMO)

Cloud Migration Orchestrator (CMO) [8] is a service system that employs BPM to provide a systematic framework to automate and orchestrate migration activities, including discovery, provisioning, network configuration, execution of migration, and validation. It is designed as a self-service portal to support migration specialists in executing migration process. CMO integrates multiple migration technologies, such as VMware vConverter, VMWare vReplication, VMWare SRM and Rackware, to support migration between different environments (e.g. VMware bare metal, xen, etc.).

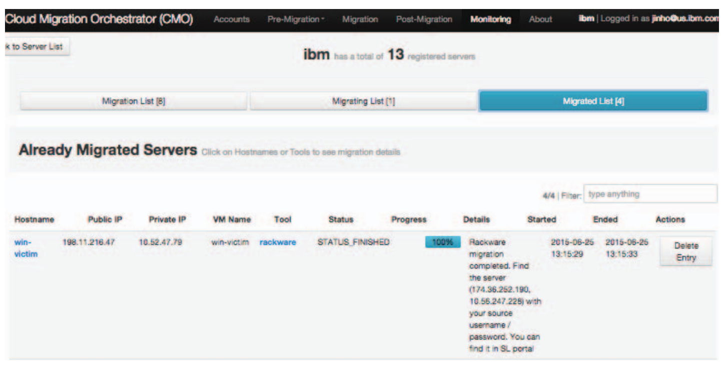
CMO facilitates any interfaces including REST API, SSH based command-line, Simple Object Access protocol (SOAP), and socket communication to automate the migration processes, and sends commands to the migration tools to initiate the migration and sends the workflow request to BPM. CMO is designed as a central point where the user controls everything without hopping to another web page or server. This way, CMO maximizes the usability and consumability.

CMO allows migration specialist to select one or more servers for migration, and a tool of choice in CMO. Also, CMO progress monitor, in Figure 2(a) provides information about the progress of each process step, such as network and target provisioning, and individual server migration status. A more detailed log is also available. Figure 2(b) shows the BPM encoding of steps involved in the pre-configuration stage, with focus on resource provisioning. Currently CMO supports migration to SoftLayer Cloud[1]. Some tasks, such as gateway provisioning are scheduled in ticketing system for manual execution by other teams.

IV. AI Planning for Cloud Migration Orchestration

This section introduces an extension to CMO, which is utilizing AI planning technology to assemble migration plans. It also presents the AI planning domain design for migration process.

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



Cloud Migration Orchestrator (CMO) | Accounts | Pre-Migration | Migration | Post-Migration | Monitoring | About | ibm | Logged in as jinho@us.ibm.com

k to Server List

ibm has a total of 13 registered servers

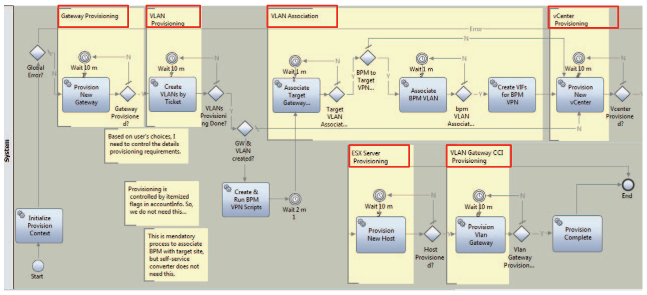
Migration List [8] | Migrating List [1] | Migrated List [4]

Already Migrated Servers Click on Hostnames or Tools to see migration details

4/4 | Filter: type anything

Hostname	Public IP	Private IP	VM Name	Tool	Status	Progress	Details	Started	Ended	Actions
win-victim	198.11.216.47	10.52.47.79	win-victim	rackware	STATUS_FINISHED	100%	Rackware migration completed. Find the server (174.36.252.190, 10.56.247.228) with your source username / password. You can find it in SL portal	2015-06-25 13:15:29	2015-06-25 13:15:33	Delete Entry

(a) CMO Migration Monitoring Screenshot



Based on user's choices, I need to control the details provisioning requirements.

Provisioning is controlled by itemized flags in accountInfo. So, we do not need this...

This is mandatory process to associate BPM with target site, but self-service converter does not need this.

Activity	Gateway Provision	VLAN Provision	VLAN Association	vCenter CCI Provision	ESX Server Provision	VLAN Gateway CCI Provision
Estimated Time	< 12 hours	< 4 hours	< 10 minutes	< 30 minutes	< 4 hours	< 30 minutes

(b) BPM Process for Automated Resource Provisioning (table shows estimated time for each step in red boxes)

Fig. 2: CMO Frontend and Backend

A. System Overview

Currently CMO employs a static BPM model of a migration workflow, leaving it to migration specialist to manually select the appropriate migration tool and handle exceptions as they arise. This section describes an extension to CMO, shown in Figure 3, to support automated composition of migration process plans. It receives a request to migrate at least one application or server image from a source (i.e., initial state) to a target environment (i.e., goal state). By employing goal-oriented inferencing from planning technology, the system relies on availability of a domain description, which captures different actions that are available and can be used to generate a migration plan. The system continuously monitors the execution of migration plan. If one or more exceptions in the execution of migration occur, the plan may be re-generated as a response to changes in the operating environment.

The current version of CMO implements "pre-configuration", "execution", and "post-configuration" stages of the migration process, shown in Figure 1. Discovery [9] and wave planning have been completed a-priori, and the results are fed into the AI planner in the form of an initial state. The target definition may be defined by a subject matter expert, who may use own expertise and any existing tools to design an optimal target environment.

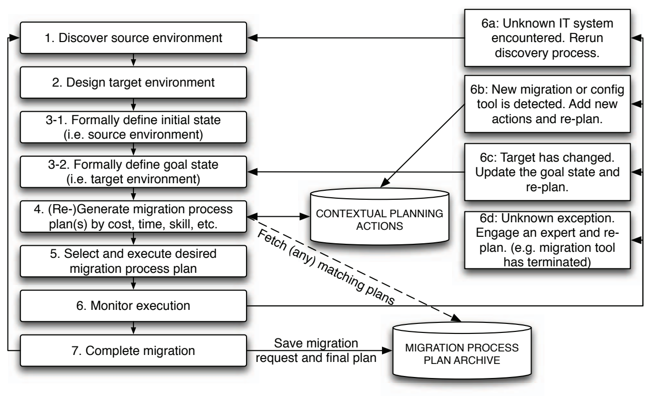

1. Discover source environment
2. Design target environment
3-1. Formally define initial state (i.e. source environment)
3-2. Formally define goal state (i.e. target environment)
4. (Re-)Generate migration process plan(s) by cost, time, skill, etc.
5. Select and execute desired migration process plan
6. Monitor execution
7. Complete migration
CONTEXTUAL PLANNING ACTIONS
Fetch (any) matching plans
Save migration request and final plan
MIGRATION PROCESS PLAN ARCHIVE
6a: Unknown IT system encountered. Rerun discovery process.
6b: New migration or config tool is detected. Add new actions and re-plan.
6c: Target has changed. Update the goal state and re-plan.
6d: Unknown exception. Engage an expert and re-plan. (e.g. migration tool has terminated)

Fig. 3: System for Automated Composition of Migration Process Plans

Learning component captures repeatable patterns in migration, for example, classified migration patterns, tooling and configuration issues, which may be common in different industries (e.g. telco or insurance workloads). A feedback loop is incorporated into the workflow for reconfiguration, and for example, to handle newly discovered items and/or properties during migration. As the migration is performed, the system may identify the "next best action" given the current context, providing an automated workflow for reconfiguration.

Migration parameters, such as operating system version on the server image, sensitivity to increased network load, cost of operations in target environment, and number of components to be migrated, impact the choice of actions that can be selected into the migration plan. One can treat them as preconditions and inputs to be met, for the action to be applicable in the given context. For example, if there is a set of similar machines, with Linux OS, which can be migrated all at once (belong to same application, or have same level of priority), SRM can be used to move them in parallel.

B. AI Planning for Migration Plan Generation

AI planning is a problem solving technique, where knowledge about available actions and their consequences is used to identify a sequence of actions, which, when applied in a given initial state, satisfy a desired goal [10]. There are three main inputs to a planner: initial state, goal state and domain description. The initial state describes the starting state of the application domain, commonly called world. The goal state describes the desired world state. The domain describes actions that, when invoked, transform the world states. The output of the planning process is a plan, a sequence of actions that can be executed in order to achieve the desired goal state.

AI planning offers an efficient means of handling multiple contexts, in the migrations domain that can be multiple source and target environments and available tooling. It promises to automate traditional manual, error-prone and tedious tasks by streamlining the decision making process.

When considering the migration process, AI planning initial state formally represents the source infrastructure, such as properties associated with servers, applications, operating

```
action name: provision_target_vmware
preconditions: [and
(and (source_vmware) (source_connected))
(or (disk_25) (disk_100) (disk_2100) (disk_4100)
        (disk_6100) (disk_8100) (disk_72000))
(or (cores_1) (cores_2) (cores_4) (cores_8)
        (cores_12) (cores_16) (cores_40))
(or (memory_1) (memory_2) (memory_4) (memory_6)
        (memory_8) (memory_12) (memory_16) (memory_32)
        (memory_48) (memory_64) (memory_512))
(or (WINDOWS_SERVER_2003_R2_STANDARD)
        (WINDOWS_SERVER_2008_R2_STANDARD)
        (WINDOWS_SERVER_2003_R2_ENTERPRISE)
        (WINDOWS_2000_SERVER)
        (WINDOWS_SERVER_2003_STANDARD)
        (WINDOWS_SERVER_2003_STANDARD_X64)
        (WINDOWS_SERVER_2003_R2_STANDARD_X64)
        (WINDOWS_SERVER_2003_R2_ENTERPRISE_X64)
        (WINDOWS_SERVER_2003)
        (WINDOWS_2000_ADVANCED_SERVER)
        (WINDOWS_SERVER_2003_ENTERPRISE)
        (RED_HAT_ENTERPRISE_LINUX_AS)
        (MICROSOFT_WINDOWS_SERVER_2008_ENTERPRISE_X86)
        (WINDOWS_SERVER_2008_STANDARD)
        (VMWARE_ESX_SERVER)
        (WINDOWS_SERVER_2008_STANDARD_X64)
        (WINDOWS_SERVER_2008_R2_ENTERPRISE)
        (WINDOWS_SERVER_2008_ENTERPRISE_X64)
        (RED_HAT_ENTERPRISE_LINUX_X64)
        (RED_HAT_ENTERPRISE_LINUX))]
positive effects: [target_vmware_provisioned]
```

Fig. 4: Sample planning action for migration using VMware converter

system. An example of initial state may be specified as follows: Server 1 is a VM on existing VMWare ESX. Server 1 together with Server 2 and Server 3 belongs to App A; Server 1 and Server 2 are Linux OS, Server 3 is Win OS. More properties may be added about the servers and their relationships and types, e.g., Server 1 is production, Server 2 is backup, Server 3 is development server, and App A is of highest business criticality level, and has the highest Service Level Agreement (SLA) in place.

Actions form the domain description. The Planning Domain Definition Language (PDDL) [11] is a common language for describing the actions in terms of input, output, precondition, and post-condition. A series of contextual actions may be defined for achieving the desired migration, each migration step based on the current context, which may be described in the input, output, precondition, and post-condition. A subject matter expert initially defines contextual actions. Migration plan is then generated. in combination with an automatic learning mechanism that learns and automatically adds new actions or modifies actions in the defined contextual actions as the system evolves. Some of the actions in the final migration plan require manual intervention, while others can be fully automated. The learning component may be able to generate new or augment existing actions based on results of other migrations. Figure 4 shows a sample action for provisioning a target on VMWare bare metal. The action ensures that source OS is of certain type, and supported by the target. It ensures that the source server exists and is reachable.

The goal state specifies the target system's configurations

```
initial-1:
    source_vmware,cores_40,memory_512,disk_4100,
    WINDOWS_SERVER_2003_R2_STANDARD,
    source_connected,userID,password,ipaddress

goal-1:
    target_vmware,source_disconnected,synched

initial-2
    source_vmware,cores_16,memory_48,disk_72000,AIX,
    source_connected,userID,password,ipaddress

goal-2:
    target_cci,source_disconnected,synched
```

Fig. 5: Sample input file with 2 initial states and matching goal states

and their properties, such as target VMs. This serves as a blueprint for what the target infrastructure and applications should look like once the migration is completed. The goal state may "mirror" the aspects of the source environment but through the lens of the migration to result in the target environment. For example, a target definition may specify to merge source servers, e.g., development server Server3 and backup server Server 2 may be merged into one server in the target environment. A formal representation for a goal state is generated. Often, this goal will be defined by the practitioner, but we propose learning-based approach to uncovering the goal, based on similar, prior, migrations. A goal state, for example, may specify that Application X be migrated to S1, S2, S3 with load balancer between S2 and S3.

The migration plan aggregates a set of actions and the context around the execution of those actions (such as how much they cost to license, the skill level of the person to execute them, the time duration associated with their execution, the risk associated with the underlying technology, and other).

Changes during execution may occur, impacting the running plan. For example, a goal set may be to migrate a server from a virtual machine to the Cloud using image migration. During execution, it has been detected that the network performance is insufficient for change window (i.e. time allocated for the server to be down for this move). At this point, a change in migration type may be triggered. In response, workload migration may be pursued. A new server may be created in the cloud. Data migration may be performed of only application. In this example, a new plan includes moving from server based image migration to workload based migration. This example illustrates exceptions shown in Figure 3, e.g., in terms of where the problem might be (e.g., source network that may be slow, intermediary network or target network).

V. Implementation and Evaluation

This section presents a set of experiments, which we performed to evaluate the suitability of AI planning for migration process automation. It also discusses technical challenges in developing the overall system and practicality of the overall approach.

A. Implementation

In our implementation we employ Metric-FF planner [12], and extension of original FF planner [13], a domain independent planning system that is based on forward chaining heuristics state based planning principles. To interface with the Metric-FF planner, we employed Planning4J framework [14], a universal Java API for connecting various AI planners. Where applicable we employed a custom mapper to transform selected actions to BPM activities, similar to [15].

B. Evaluation setup

To evaluate applicability and effectiveness of the AI planning for migration workflows, we have modeled the migration domain consisting of 10 actions and 70 predicates. In our experiments, we have focused on a migration execution stage. We assume that the IP will change in the target and that the network is provisioned already. The actions that we have modeled include migration using vConverter and Rackware, target provisioning (whether VMWare based or SoftLayer Cloud Compute Instance (CCI), sync and cutover, to name a few. The human effort in modeling the domain was approximately 6 hours, including the testing. This was based on our prior expertise with CMO and migration to SoftLayer. Figure 5 represents two sample problem definitions, which was generated based on customer inventory sheet of 700+ servers and was loaded into the planner (via Java interface).

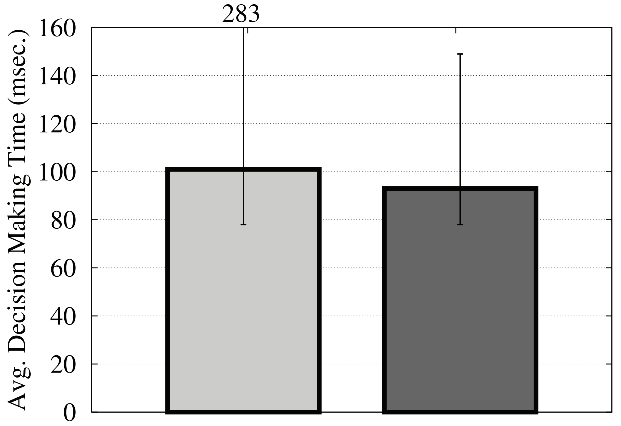
C. Experiments

The experiments were performed on a Intel Xeon CPU with 2.60Ghz, 4GB of RAM on a Windows 2007 32-bit Operating System.

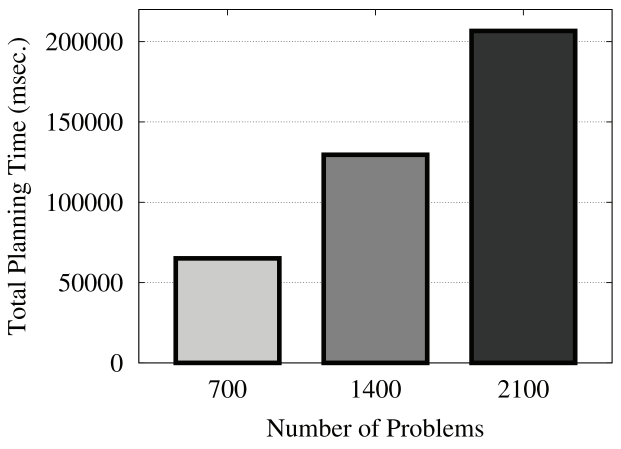
1) Scaling the number of problems.: In the first experiment, we have generated an input file with 700 problem definitions, each representing one individual request for server migration (i.e. image based migration). We have randomized some of the variables, such as, whether the source is connected or not (source machine is reachable), availability of userID and password (required by certain migration tools), and also whether OS is specified (to simulate incompleteness of data). This demonstrates realistic problems encountered during migration, such as incompleteness and inaccuracy of the discovered data. On average it took 95 milliseconds to fail a plan, and 100 milliseconds for plan to fail during search, shown in Figure 6(a). Initial state complexity ranged from 7-10 literals. Additionally we ran a planner with 700, 1400 and 2100 problem definitions, as shown in Figure 6(b) demonstrating the linearity in scaling.

2) Scaling the complexity of the goal state and initial state: In this experiment we evaluated impact of the size of the problem definition (both initial and goal state) on the performance. Firstly, we have varied the complexity of the goal state. For example, additional goals may represent scheduling constraints (e.g. if it is a production server do not migrate during the week, but only during the change window on the weekend). Performance results are shown on Figure 7(a).

Secondly, we have varied the size of the initial state, by capturing the scenarios that correspond to workload migration requests. For example, an initial state, and corresponding goal state in this experiment, may represent the request to migrate 2-instance workload or typical three-tier workload (e.g. where application X to be migrated consists of three servers). Results are shown on Figure 7(b), and as the initial state size increases there is no significant performance impact. This can be attributed to the fact that may workloads, often have a number of similar server profiles, hence the initial and corresponding state may be similar for one or more servers that belong to the workload. Experiments in Figure 7 include the average time for both successful and failed plans.



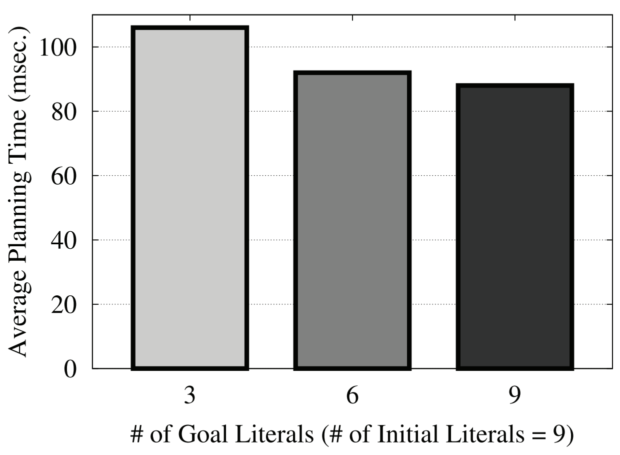
(a) Planner performance



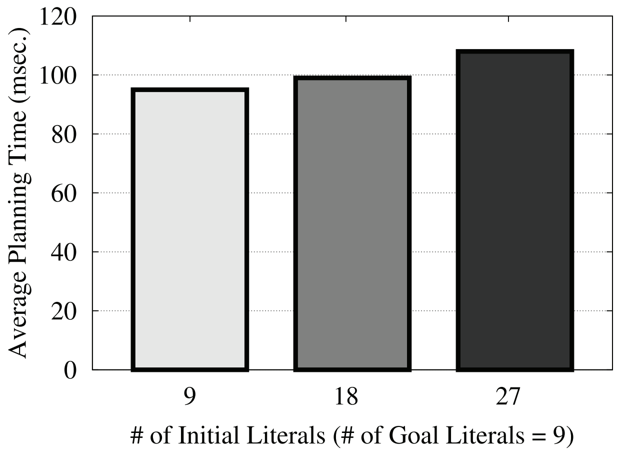
(b) Scalability of the goals

Fig. 6: Planner performance and scalability for 700 image migration requests

3) Improving the success rate of migration plan requests: This experiment evaluates failed problem definitions and applies corrective actions to improve their planning success. Firstly, for the problems of same type (e.g. VMware or CCI migration request), for each failed problem we find "similar" problem definition that was successful. We consider only problem definitions with the identical goal states. The simplest approach to deriving a scalar similarity metric from the difference between two problem definitions, is to just count the elements that they have in common. However, not all the initial states in the problem definition are equally useful

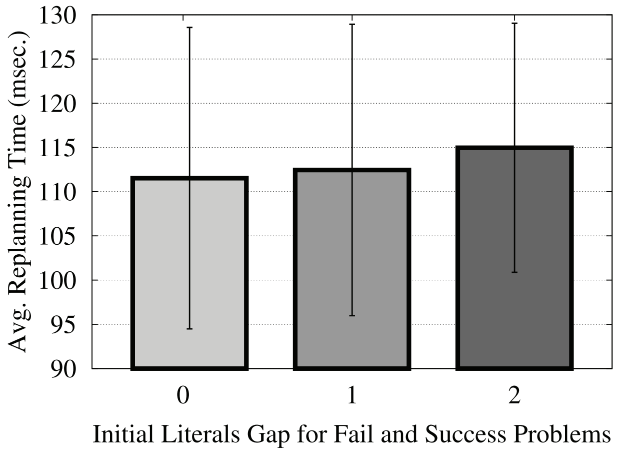


(a) Variability of the goal size

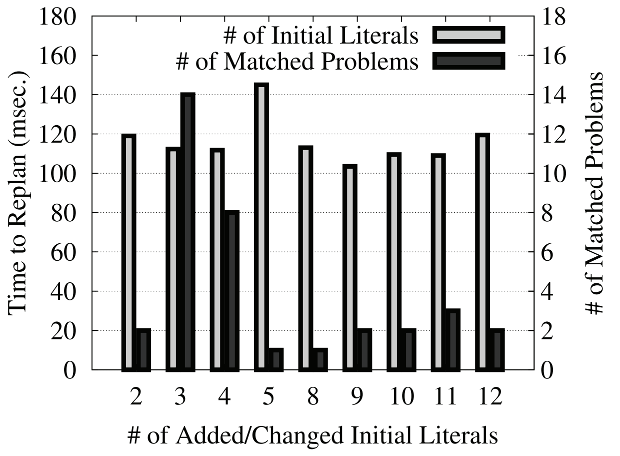


(b) Variability of the initial state size

Fig. 7: Variability of migration size request and planner performance



(a) Similarity based approach



(b) Relaxing the problem definition

Fig. 8: Improving success rates for failed problem definitions

in determining similarity. We applied domain experience to assign weights to some of the literals in the problem definition, using these as multipliers for shared elements contained in problem definitions. For example, user ID, password and the state of connectivity of the server may be ignored (to which we assigned lowest weight of 0), and give more value to the OS version and system profile, such as disk size and number of cores (to which we assigned weights as high as 5). The sub-problem evaluations were measured as value pairs *(num of literals in common x weight, total num of literals x weight)* to allow sub-states of different weights and sizes to be combined reasonably.

For similar problems (failed and success pairs), we examined the number of the literals in their description. Figure 8(a) shows there were cases where failed problems simply missed some of the literals, such as a user ID or password. This requires a human input or availability of sensing actions to gather the missing data. When both failed and successful problem had the same number of literals, the failure causes were one of the following: unsupported OS or other server configuration item, incorrect user ID or password, and disconnected source machine. Unsupported server configuration may be resolved with a proposed upgrade so that the server fits the target environment, however that incurs additional cost in migration and needs to be validated by migration specialist and/or customer. Furthermore, not all upgrade variations may be supported. Incorrect credentials require user input. Finally, unreachable server can be automatically re-started, with appropriate functions. While replanning is feasible, some adaptations to the problem definition may result in significant delays in migration itself, if human approval is required for server adaptation.

Figure 8(b) shows an alternate approach to improving plan success rates. We have randomly inserted/changed initial state literals, to simulate possible adaptations to the server configuration. In this scenario, we have extended the domain with additional actions, representing server customization options. This approach to replanning is feasible and with minimal impact to the process. It is applicable for non-critical (e.g. dev/test) servers/workloads, where human decisions may not be deemed necessary.

D. Discussion

1) Performance: The application of AI planning to manage complexity of migration workflow scenarios has been found to be usable, feasible and wellperforming. The initial domain modeling effort is minimal, when considered in the context of the duration of a typical migration process. AI planing, even if used only for simulation purposes, enables to significantly shorten this timeline, and bring forward migration errors for reconciliation. Furthermore, salespeople can use such an approach to reason about customer's environment ahead of time, and present the rough cost and time for the overall process before migration deals are signed. In current situation, as shown on Figure 1, the migration process won't start until day 70, and if any errors are encountered, the process terminates (for selected server and/or workload) and discovery and verification is restarted.

The effort involved in domain definition and modeling was 3 person days and it required expertise from migration area. Additions of new tools and new source and target environments to be supported also needs to be considered. Extension to the system, to facilitate automated discovery of new migration tools and their formalized representation in the planning domain are necessary extensions to the system, to reduce the effort in encoding the domain knowledge.

2) Workflow complexity and adaptation.: At this stage we have implemented a set of simple, atomic methods that are executed as part of the migration process. Successful plans have been translated into basic method calls, that are brokered by the intermediary server (supporting vConverter and Rackware calls). We are investigating applicable techniques to supporting more complex operators in plans, such as parallelism, and partial plans. Currently parallelism is supported on the goal level, by automatically proposing several servers (often belonging to a single workload).

3) Automation and Integration Challenges: The CMO infrastructure consists of a BPM Process Center, a BPM Process Server, a REST-based API server that also serves as a mail server for notification purposes (to include human tasks). It supports both multitenant and single-tenant models. In the multitenant model, the global environment supports multiple migration orchestration instances serving multiple customers. In the singletenant model, the orchestration infrastructure is replicated to the customerś target site, serving only the said customer.

To fully streamline and automate our approach for the entire process, there are a number of technical challenges to overcome:

- Support for automated reasoning about new migration tools to generate corresponding domain actions
- Support for the different interfaces: Not all the tools provide REST API. For example vConverter supports SOAP only, while Rackware provide only CLIs.
- Support for the different migration flows: Flows vary based on the source, target and context of operations. This is where re-planning and interleaved planning and execution offer a solution.
- Availability of a common, unified data model for all migration tools: Tools have different input requirements, data formats and prerequisites for migration.
- Ease of tool maintenance and aggregation of composite log messaging across migration workflows.

E. Goal inferencing

At present the system receives a migration request, which is translated into a formal system goal. This opens up a space for novel approaches to automated goal inferencing based on the input, source environment and available target environments. Ideally the system should define a goal that transforms and optimizes the image/workload in the target environment.

VI. Related Work

This section puts our work in the context of related research in BPM for cloud services, applications of AI planning for IT service management, an applications of AI planning techniques for adaptive workflows.

Applications of BPM in cloud environments introduce a number of challenging issues at design and runtime. These include, application developers seeking flexibility (to enable service integration from multiple providers), process owners that are interested in cost reduction, while offering a context-aware, comprehensive solution, and users who require performance. To address some of these issues, Muthusamy et al. [16] present service level agreement (SLA) approach to business process management for service oriented application in cloud environments.

Schulte et al.[17] present an architecture for an elastic Business Process Management System, and discuss a number of challenges in this domain, including scheduling and resource allocation, process monitoring, and state management; but do not explicitly call out support for context-awareness and adaptation.

Keller et al. [18] focus on change management process, which tackles the changes and upgrades of IT systems (whether software or hardware ones). They introduce system called CHAMPS, designed to assesses the impact of the change and generates a change plan (as a BPEL workflow). CHAMPS employs AI planning to optimize selection of resources and the execution time, yet the provisioning of operation and their temporal dependencies are pre-defined.

In contrast to [19] our work utilizes a multi-variable approach to selection of tooling and services to be executed during migration. The plans and next actions are selected based on a number of constraints, such as cost, duration and skill required. Furthermore system embeds a learning component that captures repeatable patterns in migration (classified migration patterns, tooling and configuration issues) to replicate migration use cases over the time.

AI planning has been investigated as means for automated Web service process composition and adaptation in a number of efforts [20], [21], [22]. In our prior work [23] we applied AI planning for context aware service composition, and subsequently have proposed a system for goal transformation [24], if the plan generation failed. That work, however does not tackle the pattern learning and goal inferencing for process planning.

VII. Conclusion

In this paper we demonstrated a (self–service) system for cloud migration process orchestration, which employs AI planning to automate migration workflow generation and recover from failed plans during the execution. We demonstrate effectiveness of the AI planning, as means of improving process performance in today's migration approaches. We discuss our insights from running AI planning to generate workflows for 700+ image migration requests. We discuss a set of technical and automation challenges that we have encountered while developing our solution.

In more general sense, we observe the requirements for the next generation of IT service management systems to be extremely flexible and scalable processes across hybrid Cloud environments. As the enterprise IT continuously evolves, it is key that these processes are adaptive, and AI planning offers an approach to achieving flexible composition and recovery from failures. As a future work, we will pursue two directions. Firstly, evaluation of learning approaches to goal-inferencing and pattern-based planning. Secondly, we will investigate governance challenges and how to generate adaptive process workflows based on services that are offered by diverse providers (e.g. migration technology or network providers).

References

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] W. M. P. Van Der Aalst, A. H. M. T. Hofstede, and M. Weske, "Business process management: A survey," in *Proceedings of the 2003 International Conference on Business Process Management BPM"03*. Springer-Verlag, 2003, pp. 1–12.

[3] C. Pautasso, O. Zimmermann, and F. Leymann, "Restful web services vs. "big"' web services: Making the right architectural decision," in *Proceedings of the 17th International Conference on World Wide Web*, ser. WWW '08. ACM, 2008, pp. 805–814.

[4] M. Hajjat, X. Sun, Y.-W. E. Sung, D. Maltz, S. Rao, K. Sripanidkulchai, and M. Tawarmalani, "Cloudward bound: Planning for beneficial migration of enterprise applications to the cloud," *SIGCOMM Comput. Commun. Rev.*, vol. 40, no. 4, pp. 243–254, 2010.

[5] E. Keller, S. Ghorbani, M. Caesar, and J. Rexford, "Live migration of an entire network (and its hosts)," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, ser. HotNets-XI. ACM, 2012, pp. 109–114.

[6] J. Zhang, L. Renganarayana, X. Zhang, N. Ge, V. Bala, T. Xu, and Y. Zhou, "Encore: Exploiting system environment and correlation information for misconfiguration detection," in *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '14. ACM, 2014, pp. 687–700.

[7] M. Nidd, K. Bai, J. Hwang, M. Vukovic, and M. Tacci, "Automated business process discovery," in *IM'15*, ser. IM'15, 2015.

[8] J. Hwang, Y.-W. Huang, M. Vukovic, and N. Anerousis, "Enterprise-scale cloud migration orchestrator," in *IFIP/IEEE IM 2015 Symposium*, 2015.

[9] J. Jermyn, J. Hwang, K. Bai, M. Vukovic, N. Anerousis, and S. Stolfo, "Improving readiness for enterprise migration to the cloud," in *Proceedings of the Middleware Industry Track*, ser. Industry papers. New York, NY, USA: ACM, 2014, pp. 5:1–5:7.

[10] M. Ghallab, D. Nau, and P. Traverso, *Automated planning: theory & practice*. Elsevier, 2004.

[11] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "Pddl-the planning domain definition language," in *The International Conference on Artificial Intelligence Planning Systems (AIPS-98) Planning Competition Language Specifications.*, 1998.

[12] J. Hoffmann, "Extending FF to numerical state variables," in *Proceedings of the 15th European Conference on Artificial Intelligence (ECAI-02)*, Lyon, France, Jul. 2002, pp. 571–575.

[13] J. Hoffmann and B. Nebel, "The FF planning system: Fast plan generation through heuristic search," *Journal of Artificial Intelligence Research*, vol. 14, pp. 253–302, 2001.

[14] M. Cerny, "Planning4J - Java API for AI planning. available at http://code.google.com/p/planning4j," 2012.

[15] J. Hoffman, I. Weber, and F. M. Kraft, "SAP speaks PDDL: Exploiting a software-engineering model for planning in business process management," *arXiv preprint arXiv:1401.5858*, 2014.

[16] V. Muthusamy and H.-A. Jacobsen, "BPM in cloud architectures: Business process management with SLAs and events"," in *Proceedings of the 8th International Conference on Business Process Management*, ser. BPM'10, 2010, pp. 5–10.

[17] S. Schulte, C. Janiesch, S. Venugopal, I. Weber, and P. Hoenisch, "Elastic business process management: State of the art and open challenges for {BPM} in the cloud," *Future Generation Computer Systems*, 2014.

[18] A. Keller, J. L. Hellerstein, J. L. Wolf, K. lung Wu, and V. Krishnan, "The CHAMPS system: change management with planning and scheduling," in *Network Operations and Management, IEEE Symposium*, vol. 1, 2004, pp. 395–408.

[19] K. El Maghraoui, A. Meghranjani, T. Eilam, M. Kalantar, and A. V. Konstantinou, "Model driven provisioning: Bridging the gap between declarative object models and procedural provisioning tools," in *Proceedings of the ACM/IFIP/USENIX 2006 International Conference on Middleware*, ser. Middleware '06, 2006, pp. 404–423.

[20] P. Bertoli, M. Pistore, and P. Traverso, "Automated composition of web services via planning in asynchronous domains," *Artificial Intelligence*, vol. 174, no. 3–4, 2010.

[21] R. Akkiraju, K. Verma, R. Goodwin, P. Doshi, and J. Lee, "Executing abstract web process flows," in *In Proceedings of the ICAPS Workshop on Planning and Scheduling for Web and Grid Services*, 2004, pp. 9–15.

[22] A. Gonzlez-Ferrer, J. Fernndez-Olivares, and L. Castillo, "From business process models to hierarchical task network planning domains," *The Knowledge Engineering Review*, vol. 28, pp. 175–193, 6 2013.

[23] M. Vukovic and P. Robinson, "Adaptive, planning based, web service composition for context awareness," in *Proceedings of the 2nd International Conference on Pervasive Computing*, 2004.

[24] ——, "Goalmorph: partial goal satisfaction for flexible service composition," in *International Conference on Next Generation Web Services Practices*, 2005.