Ambient calculus for distributed, service oriented virtual machines

- Context

Service oriented architectures are new ways of designing applications through components and service interactions. Applications are described using some Architecture Description Languages (ADL). A specific execution framework brings together pieces of this architecture, starts a main component and enables interactions between these components. We are considering ambient environments where components have complex life-cycles and where applications ADL are not stable. Old components can be dynamically substituted by more recent ones, components can be discarded if they consume too much resources or components can disappear if they are out of network range, etc.
Most of the time the ADL is used to guarantee a good expression of the application, We want to extend this approach, to include various guarantees when the entire ADL is not always resolvable, ie when all components of the ADL are not immediately available.

Classical approaches use XML as a way of expression application ADL. We think that using formal models such as the ambient-calculus to express application ADL will help to control application characteristics. Such formal models can be used to control compatibility of a component evolution and its associations with other ones while the application changes during time.

- Approach

Our approach consist in first establishing an application ADL based on the ambient-calculus. Then, we intend to map this ambient-ADL to the run-time architecture (in particular a java service oriented framework). Finally we wish to apply calculus based on application behavior when changes occur and control that the overall application is still coherent.

- Expressing application ADL: The task here consists in establishing an ambiant-calculus expression of existing applications. Two approaches are possible: we can either start from an existing ADL to express desired properties, or we can try to build the expressions from running applications. We propose to analyze log information from running applications and find ambient-calculus expressions (abstractions) of their behavior.
- Ambient-calculus mapping to specific run-time: two architectures are targeted in our approach: webservices and OSGi. Webservice applications are mostly focused on data exchanges through service calls and are good candidates for ambient-calculus expressions, but they lack execution context integration. OSGi is not a distributed system but it is targetted at dynamic ambient environments and it follows a service-oriented programming paradigm. We propose to focus on the OSGi framework and find translations between ambient-calculus terms and OSGi service oriented paradigms.
- Ambient-calculus expression fixed point: ambient-calculus expressions must help us to analyze application stability over changes. For instance when new components are installed, it must control that newly observed behavior has the same ambient-calculus external representation as the previous one with regards to the application description. It can also control that if a new component arrives with specific constraints it is still compatible with the initial specification.

Next figure shows how ambient calculus could be used on a OSGi run-time.

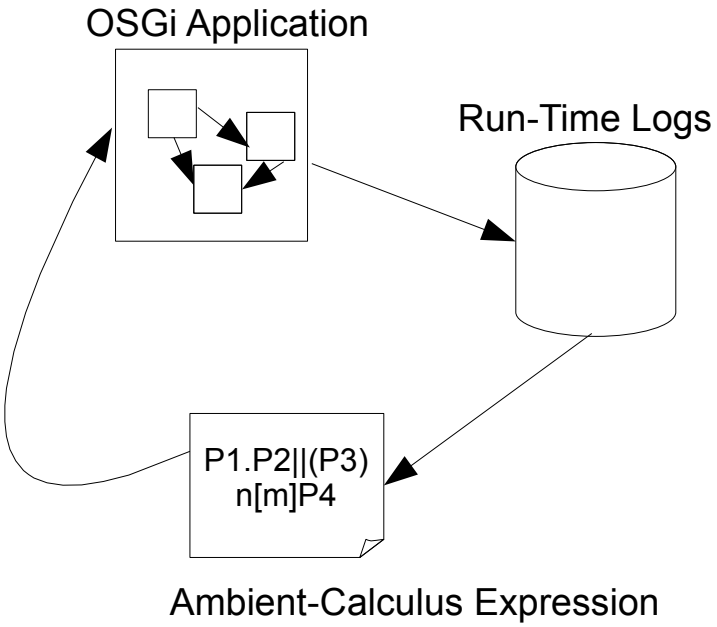

Figure 1: Ambient-Calculus Expression extraction from run-time logs

The expressions are extracted from the OSGi log analysis. We use the calculus to find the simplest log expression and reuse it within the application to guarantee stability even-though components are changed.

- Thesis Organization

The phd candidate will have to study ambient-calculus and its variations, map this kind of expression to the service oriented programming world, and find expression controls that improve application stabilities over time.

分布式，面向服务虚拟机的环境演算

- 背景

面向服务的架构是设计应用的新方法，它通过组件和服务之间的交互来完成。应用通过某些体系结构描述语言来描述。特定的执行框架把这种体系结构的各个部分结合起来，始于一个主组件，并能在这些组件间完成交互。考虑组件有复杂生命周期的环境，在这种环境下不适合使用 ADL。老的组件可以动态地被更新的组件替代，如果组件消耗了太多的资源，这些组件可以被丢弃，或者如果组件在网络范围之外，这些组件就自动消失了。大多数情况下 ADL 用于保证很好地描述应用。但当整个 ADL 并不总是可解时，即 ADL 的所有组件不能立即有效时，需要扩展这种方法来提供各种不同的保证。经典的方法是使用 XML 来表示 ADL 应用。使用形式模型，如环境演算，来表示 ADL 应用，将有助于控制应用特性。当应用随着时间变化，这样的形式模型可以用于控制组件进化的兼容性及组件与其他组件的关系。

- 方法

我们的方法是首先基于环境演算建立 ADL 应用，接着，映射环境-ADL 到运行体系结构（尤其是面向 Java 服务的框架），最后，当发生变化时，应用演算并控制整个应用保持连贯性。

•ADL 应用表示：建立应用的环境演算表示。有两个可行方法：或者使用已有的 ADL 来描述需要的特性，或者从运行应用程序中建立这种描述。我们提出分析运行应用程序的日志信息的方法来建立应用程序行为的环境演算抽象表示。

•环境演算：映射到运行环境，我们考虑两种结构，Web service 和 OSGi. Web service 应用数据交换是通过服务调用，对于环境演算表示是一种好的选择，但缺少执行上下文集成。OSGi 不是分布式系统，但它适于动态环境，而且它遵循面向服务编程范式。我们提出使用 OSGi 框架，并对环境演算表示和 OSGi 面向服务范式进行转换。

•环境演算表示要点：环境演算表示必须能帮助我们分析应用在变化中的稳定性。例如，当安装了新的组件，环境演算必须能控制新观察到的行为要有与之前相同的环境演算外部表示。它也要保证如果带有特定约束的新组件加入，仍能和之前的规范兼容。

下图演示了环境演算如何用于 OSGi 运行环境，从 OSGi 日志分析中提取表示。我们使用演算来找到最简单的日志表示，并在应用中重利用来保证稳定性，即使组件发生变化。

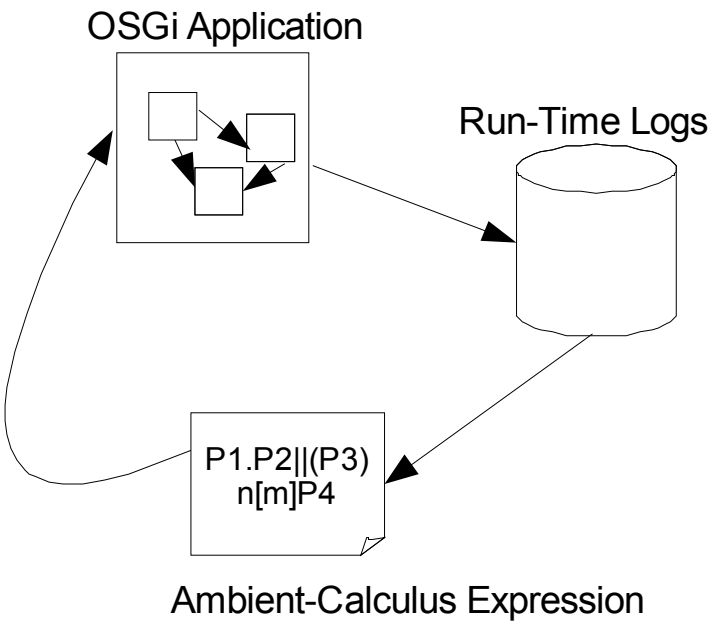

图 **1**：从运行日志中提取环境演算表示

- 论文组织

博士生将研究环境演算及其变化，映射这种表示到面向服务编程的方法，提高应用随时间推移的稳定性的表示控制方法