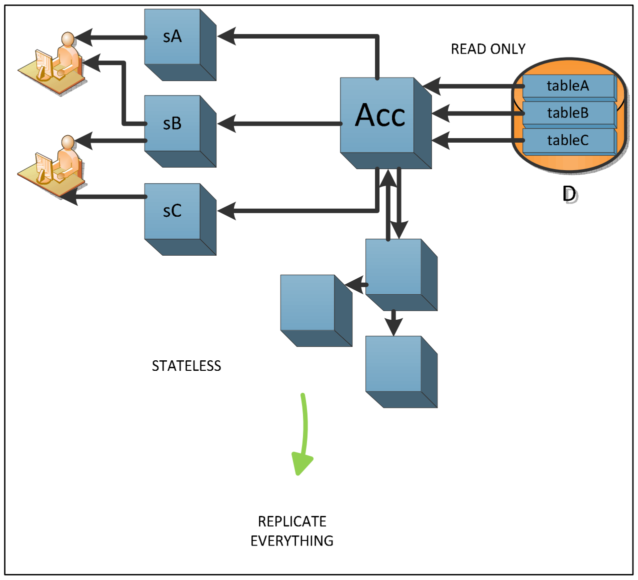
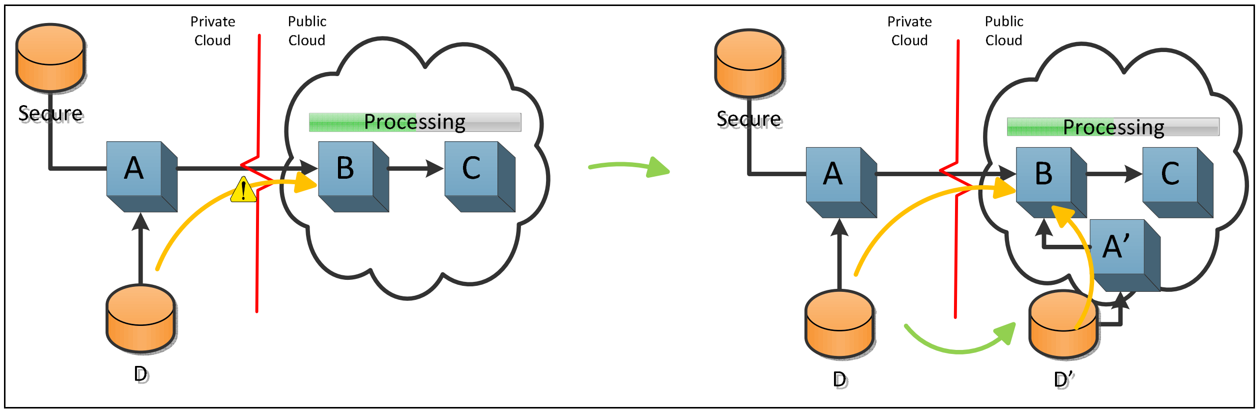
Thesis Statement

Problem:

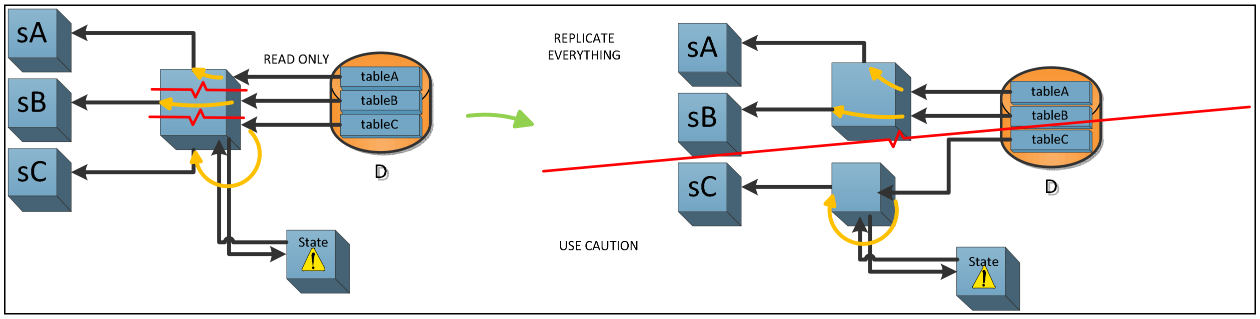
When migrating an application to a new environment - in particular when using cloud deployments, dealing with large amounts of data, and seeking to parallelize – it is very useful to know certain things about an application's components. I seek to provide an analysis which identifies opportunities for replication of computation and data access as well as possible restructurings of an application to give more freedom to how its data is accessed – information which such current tools as application partitioning frameworks largely do without.



My analysis would need to indicate that such an application as depicted above uses only read-only data, and keeps no state between user requests to it. As such, one could replicate the application freely.



In this case, say that a partitioning scheme would aim to place execution of component A privately due to interaction with secure data, and components B and C in the public cloud to offload computation burdens. Such a scheme may be undesirable, however, because of communication costs between A and B. My analysis would need to detect and indicate the viability of moving data from D to D' in the public cloud, and allowing B to access it through replicated functionality from A, A'. Knowing which components depend on this data, and if the A' functionality can be moved like this is critical.



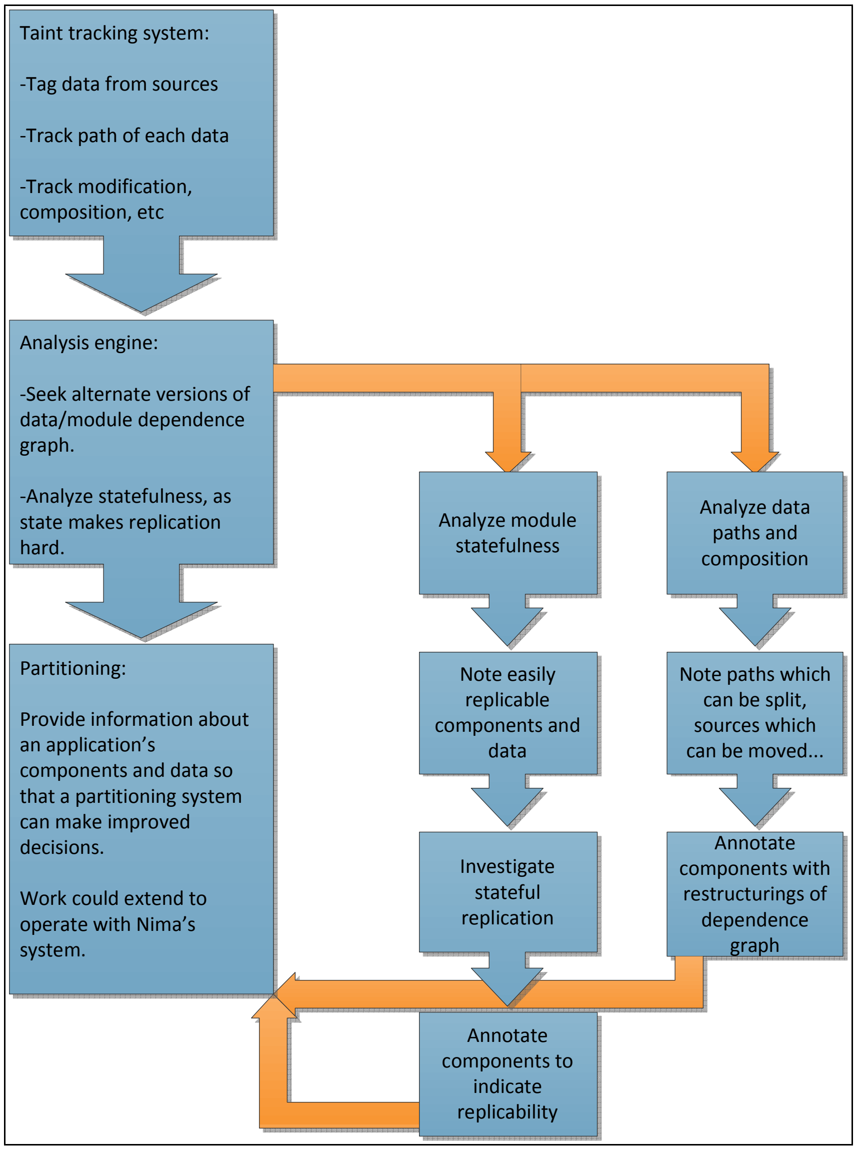
This last example has two important aspects. First, even though the servlets sA, sB, and sC all use the same accessor component (which could comprise many classes), the data flow from their respective tables to servlets does not mix. Second, in the case of sC the communication is stateful, with data being stored somewhere (in memory) between executions of sC. Knowing these things, one could then freely replicate the sA and sB parts of the system, separating them from the sC portion where greater care must be taken.

These examples are not exhaustive, but serve to demonstrate the variety of even simple cases where collecting this kind of data can greatly empower other tools. The components of an application could be annotated with such data, available for tools like partitioners to use to produce better results.

Hypothesis:

I propose that dynamic taint tracking is a good fit to the problem of gathering this kind of data. It can be used to observe how data moves through an application, how it is modified and composed with other data, and how it is persisted. Such data from taint tracking can be analyzed to determine the safety and potential costs of replicating various execution and data components. It can also be used to reason about restructuring an application with respect to how it accesses its data. By annotating components with this information, a tool such as a partitioning framework can make better decisions.

The following diagram illustrates the proposed system:


Taint tracking system:
-Tag data from sources
-Track path of each data
-Track modification, composition, etc
Analysis engine:
-Seek alternate versions of data/module dependence graph.
-Analyze statefulness, as state makes replication hard.
Partitioning:
Provide information about an application's components and data so that a partitioning system can make improved decisions.
Work could extend to operate with Nima's system.
Analyze module statefulness
Analyze data paths and composition
Note easily replicable components and data
Note paths which can be split, sources which can be moved...
Investigate stateful replication
Annotate components with restructurings of dependence graph
Annotate components to indicate replicability

Evaluation:

To demonstrate these claims, first an appropriate taint tracking system will be built using Aspect J. This system will need provide detailed data on any Java web application concerning how it interacts with data. This will involve tagging data as it is read from databases and input from users, and tracking it as it is wrapped in objects, modified, composed, and written.

The output of this stage should be clear graphical representations of an application's use of data, such that one can begin to get an idea of how components might be replicated and/or restructured.

Following the taint tracking, the next major goal will be the development of the analysis engine, the output of which will be automatically generated statements about the replicability of components and viable application restructurings (eg. an accessor whose responsibilities can be split across hosts, or moving data closer to the component which uses it by moving the data and the code execution needed to access it).

Such can be evaluated by creating a series of test applications about which these properties are known, and having my system correctly determine them automatically. A more involved approach could be to analyze an existing application with my system, and then study the process of acting on its suggestions.

Finally, if it is determined to be in scope of this project, it would be interesting to evaluate integration with Nima's partitioning framework. His algorithms would need to be modified to act on the data I provide, to see if superior partitionings can be produced compared to what the algorithms currently do.

Timeline:

Taint tracker:

-Tracking string/object taint from databases accesses.
-Tag data items with sources and keep track of these (this will track composition)
-Detect data modifications
-Extend tracking to user input
-Extend tracking to other data types, if possible (notably primitive arrays)
-Clean output format
-Demonstration graphs for JGossip and another application

Expected Completion Date: Mid February

Analysis engine:

-Find a means to characterize state
-Engine identifies stateless component execution
-Engine identifies easily replicable data
-Engine identifies stateful component executions and associated data
-Engine provides data to help evaluate cost of replicating stateful components
-Engine identifies independent flows of data, even when state is involved
-Engine identifies components which can be split up into sub-components
-Engine identifies data sources with freedom of placement

Expected Completion Date: Late March

Evaluation:

-Analyze JGossip and another application, manually check results
-Develop a series of simple test applications with known characteristics that analyses should find
-Find a suitable existing application to analyze and attempt to replicate/restructure
-Evaluation could also include the following Integration stage of the project

Expected Completion Date: Late April

Integration:

-Note that this is a non-essential step. The tool will be designed in such a way that such integration is possible, but actually doing this is not necessary for the validation of the research.

-Present data ('annotate components') in a format Nima's tool can easily use

-Consider analyses which take data from Nima's tool as further input
-Find a suitable application to partition with added data
-Search for better partitionings
-Attempt to enact such a partitioning

Expected Completion Date: Early April