Addis Ababa Institute of Technology

Distributed Nodal Processing System

Software Architecture Proposal

Samuel Sisay

Concept

Since the dawn of the computing era, hardware and software always have been in a race for accommodation. Hardware is built and improved at breakneck speeds to allow for greater performance and effectiveness of existing software, and software is continuously modified and developed so as to maximize the utilization of existing hardware. Up until the spread of distributed systems, this relationship has been the cause of many hang ups and bottlenecks in the computing industry. Distributed systems allow for greater performance and sharing of hardware resources, but architectural styles that currently follow this design tend to nail down the applications as fixed services. This means a lot of effort and time goes into development of these services in order to facilitate reusability for a multitude of other applications. This is all very well should this be a common run of the mill software that uses conventional and mundane resources to do unextraordinary tasks, but for experimental and unorthodox developments it is an expensive risk.

In addition, distributed systems stress separation of concern in development. This usually entails usage of different frameworks, languages and sub-architectures, which, although highly beneficial eliminates the possibility of packaging everything into one application. Not much flexibility or speed in development exists when dealing with such systems. So, we have our issue to solve, lack of speed and flexibility in development of distributed systems.

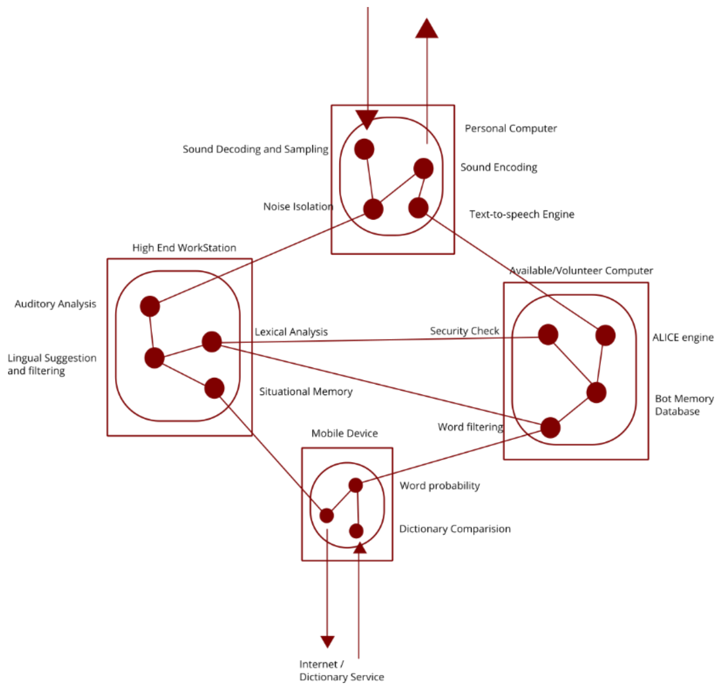

Fig 1.1: A DINOPS implementation of a Speaking Chatbot

The DINOPS architectural style is essentially the node graph sub-architecture prevalent in the graphics industry applications adapted to function as a form of peer to peer architecture. By categorizing different functions of an application as nodes and distributing them throughout a set of hardware, we get to exploit hardware functionality, maximize software maintainability and allow easy and speedy development for distributed systems.

Components

A DINOPS system must contain the following three things.

1. Nodes
2. Pipes/Links
3. Clusters

Nodes are the basic units of such systems. They are computing sections of the software of which the defining function is to run a service that receives data from one or more nodes, processes and passes on the results to other nodes. These nodes are sub-applications that are continuously run, sharing the same or on different devices as other nodes. A set of nodes can be distributed throughout many devices as the developer sees fit.

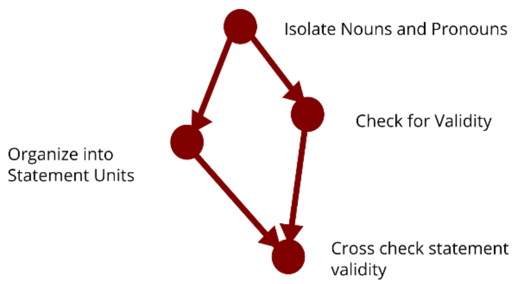
Isolate Nouns and Pronouns
Check for Validity
Organize into Statement Units
Cross check statement validity

Fig 2.1: Node examples for Language interpretation

Pipes are the pathways through which data flows through nodes. The concretion of these objects instead of having the nodes communicate directly with each other allows for separation of concern in the building of nodes, to have a simple focus on the computation functions that they must carry out. Pipes are concerned with the layout of the application and make data stream synchronization possible which is an issue in branching systems. Additionally, insertion, removal and modification of node location and hardware is simplified when using pipes.

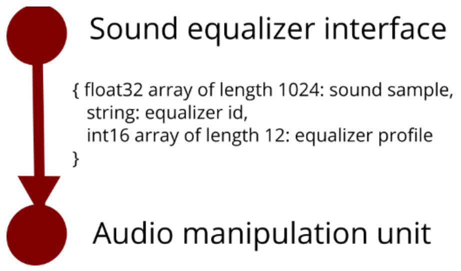
Sound equalizer interface
{ float32 array of length 1024: sound sample,
string: equalizer id,
int16 array of length 12: equalizer profile
}
Audio manipulation unit

Fig 2.2: Pipe examples for Audio editing

Clusters are the organization method of nodes and an abstraction for the device that runs the set of supplied nodes. A series of nodes running on a single device as part of a larger system form a cluster, and DINOPS systems are composed of clusters that have nodes operating within them. Clusters also contribute to the separation of concern by handling the issues of networking and providing an environment for the nodes that allows for simple communication with other nodes, be they in the same or different cluster. That means to a node, there is no concept of network distribution or any such hurdles. All other nodes operating for the system are within arm's reach. It is also possible to use clusters as a tool for categorization of nodes via function.

Design

In a DINOPS system, an application is at minimum a single cluster with two nodes. One of these nodes would serve as an input portal to the system where data is fed. The other node would be the retrieval point, where data processed by the system would be available for pick up. The difference of this architecture from the pipe and filter architecture would be the availability of multiple input and output points from the system. Instead of a single pipeline through which data passes, there are multiple paths interweaved among each other possibly headed to different output points. This allows for reuse of data from a previous point in the path for calculations down the road. It also allows for less overhead, since data is directed to the location that it is needed, instead of travelling with everything else in a big bulky singular stream.

When nodes are added into a cluster, they are connected to the nodes that they would like to receive data from, and the nodes that they will be delivering their output to. With the addition of more hardware, the functionalities that that hardware would be performing would be specified in a series of nodes encompassed in a singular cluster, that would be run on their respective hardware.

Procedure

When an application is developed that implements this architecture, the best way to go about it is in the following way.

Functional Design

The application needs to be decomposed into segments. These segments need to have a clear function within the system, and although the inner workings of the code is not essential to the architecture, the types and amounts of data that the segments would need to operate need to be clearly defined.

Nodal Design

Once there is a clear understanding of what segments are needed to build the application, these segments need to be transformed into nodes. The links between these nodes need to be established keeping in mind the data types that they would be transferring and finding optimal routes for processing. In designing the nodes, it is important to understand that as beneficial as they are to keeping the systems functionality organized, an excess of nodes would mean that the data would have to go through more pipes, and more time would be spent in transit when it should be spent in processing. Therefore, there should be a balance between distribution and compactness when designing the nodes.

Cluster Design

How the nodes are positioned in clusters has a very significant impact on the performance of the system, since this is where hardware capabilities and network streams come into play. There are two main things to consider in deciding which clusters to group nodes into.

Hardware

Since a cluster is run on a device, steps should be taken to ensure that the nodes within that cluster would perform better on that device than any other in the system. For example, if we have a series of nodes that use CUDA operations in their calculations, and there is a device with a very capable GPU available on the system, those nodes should be put into the cluster of that device. Similar with all other nodes, it is important to optimize the hardware and software within the system to get maximum performance. This is one of the main strengths of the DINPOS architecture, and must be paid attention to.

Networking

Clusters are the applications that run on their respective devices, and even though they simulate a single environment for the nodes to work in, nodes on different clusters are still on different devices, so data communication between them relies on the network connection. This means if there are nodes communicating with multiple other nodes in other clusters, the data streaming speed is bottlenecked by the connection speed, hence the overall performance of the system. In order to decrease the effect of network speed on the system, the nodes should be grouped in such a way communication between clusters is minimized. When the more node communication takes place within the bounds of a cluster, the faster the data transit, and the better the performance of the system.

There are times in which it is not possible to have these two conditions fulfilled without contradicting each other. For instance, there may be a system which have nodes that when put in their optimal hardware utilization clusters, result in excessive and inflated data streams between clusters, and when rearranged to minimize cluster communication, end up with poor hardware utilization. On such occasions, it is important to have a compromise, in order to get the best out of the system.

Application

The DINOPS architectural style is applicable in any setting for the development of any type of software application, whether it is a small or large one. However, like all software architectural styles, it has situations in which it shines and ones to which it proves more of a hindrance than help.

The best use of this architectural style is in systems that require very heavy computation and large data transfer, such as big data systems. The strength of this architecture is very well displayed in systems that have massive amounts of data that needs to be processed in multiple different ways. As a node based system, it allows for many paths of data movement, which is recommended for systems that use varied simultaneous processing.

Optimally, a DINOPS architecture should be used in a system that processes large streams of data with varied and non-uniform computations. Farms.

Assessment

Using this architectural style has its benefits, however it is not without drawbacks. From a theoretical standpoint, here are a few of the effects that would be observed in a system that implements the DINOPS architecture, both positive and negative.

Advantages

- Decomposable. Systems are built in a way that is easily understandable and modifiable. Clearly defined categories of function.
- Scalable. System size can be boosted by addition of clusters and nodes, with minimal modification to the existing structures.
- Lightweight. As far as distributed systems go, this architecture allows for building of lightweight yet broad network-based systems.

- Available. Through redundancy in nodes and clusters, a DINOPS system can be configured to withstand interruptions that may occur.
- Small Systems Friendly. It allows for the development of distributed systems without large servers but instead large coagulations of small or low spec computers.

Disadvantages

- Performance Threshold. As with most big data computation systems, the performance advantage is not exactly visible until the amount of data that should be processed exceeds a certain threshold.
- System type. Not very useful when developing data retrieval and display systems such as websites and apps.
- Reusability. The architecture encourages development of short term specialized units to build systems, so not as many reusable units are produced.
- Untested in Big Systems. This architecture has not been tested in large scale systems so there are unknowns in that area.