agus@orbleaf.com

Orb-Weaver

if the radiance of thousand suns were burst at once into the sky
that might be the splendor of mighty one.

Introduction

Different platform require different programming environment to optimize their advantages over others, C language makes up 60% of entire embedded system development, for desktop programming there are Java and dotNET, SQL query to interface database, R for statistical analysis, PHP for server side and JavaScript for client side.

The problem is that most of programmers are only specialized between 2-4 languages in his lifetime; making switching to different type of platform difficult. Orb-Weaver is a virtual machine that can be ported to different type of platform, virtual machine provides the necessary abstraction on hardware level for machine byte codes to run on different type of microprocessor architecture, this kind of mechanism guarantee that each byte codes would be compatible across platform.

To interface Orb-Weaver virtual machine with higher level programming language a compiler is needed for this purpose, the current glome compiler introduce Orb-Script language a language derived from ECMA262 (for easy migration process), it is designed for processing and manipulating data and control program flow, therefore it doesn't offered algebraic axiom for complex mathematical operation (though we can do this through APIs).

Orb-Script/Orb-Weaver are designed for scalability and interoperability between machine and programming APIs, the Orb-Weaver virtual machine is capable to be ported on constrained machine such as smart card, Orb-Script Itself offered a collaboration mechanism through use of system APIs and framework that didn't exist in ECMA262 specification, this kind of mechanism enable Orb-Script/Orb-Weaver to provide high level abstraction for different type of platform.

Orb-Weaver

Orb-Weaver is small compact virtual machine designed to fit within a constrained devices such as smart card while maintain the scalability for different type of platform through system APIs, Orb-Script would provide the necessary framework to interface this APIs on programming scale. the design philosophy based on 6-S features.

- Small, intended for constrained device therefore code size and memory usage needs to taken into account.
- Stack, it is based on stack machine for small memory footprint.
- Seamless, communication between system APIs, framework and user code could be done in seamless way.
- Scalability, functionality can be expanded for different type of purpose and hardware platform.
- String, all data are treated and manipulated as sequence of bytes.
- Secure, applications are installed/deleted in secure environment; it cannot be copied outside the host device.

With all data treated as sequence of bytes (struct vm_object) enabling different type of data representation (array/object, string, context, instance) to be processed by the compiler, type checking was done on system APIs layer (an exception will be generated when processing an invalid data) at runtime.

During the design phase back in 2014, the reason to use virtual machine instead of native code generation, is that some hardware such as 8051 microprocessor and most of 8 bit microprocessor doesn't support re-locatable code as it's still use absolute addressing for code execution, therefore it eliminate the possibilities for multi-application installation.

There are total 24 op-code making up the total instruction set for virtual machine, by reducing the total number of op-codes, it reduce the code size for ported hardware codes (a typical 8051 only cost 26KB of flash memory for entire decoder and system APIs) and making portability to different platform easier (also ported to C#, and future also can be ported to another platform such as JavaScript for client execution), this is possible because the virtual machine didn't implement advanced mathematical operation for processing floating point and other advanced logic/arithmetic unit.

Automatic Garbage Collection

Because orb-weaver running on constrained devices a significant measurement must be taken to reduce the memory usage during execution, and automatic garbage collection based on reference counting is used for this purpose. The virtual machine will automatically collect and release any heap memory for variable in-case the variable is not referenced anymore, the process are done in the background and user won't be able to realized it. On another platform such as dotNET or JavaScript, garbage collection are already handled by their runtime, therefore porting the virtual machine to these platform require no implementation of garbage collection.

Automatic Exception Handler

Most of Orb-Weaver type checking are done during runtime (the compiler didn't provide any type checking as it treat all variable as object), and automatic exception handler are employs for this purpose, automatic exception handler will terminate execution in-case user program running outside the bounds of virtual machine.

Ported Platform (and Future Platform)

Orb-Weaver is currently ported to several platform, either hardware or software. It offered same programming paradigm through different platform, eliminates the need to understand the low-level architecture of each platform.

Smart Card Platform (Orb-Weaver Card)

The first Orb-Weaver is designed to run on smart card device, it is intended to control the program flow and provide mechanism for data manipulation/processing for ETSI102.223 and 3GPP 11.14 specification known as SIM Toolkit Application (the technology behind mobile banking). The current de-facto platform (since 2001) for building SIM Toolkit Application is SmartTrust WIB (native development also possible at increased complexity based on state machine), the problem with SmartTrust WIB is the programming language still based on markup language (called Wireless Markup Language or WML) and no scalability, each SIM Toolkit API are defined literally within the language as part of it's syntax, also without a significant knowledge in ETSI102.223/3GPP11.14, developer couldn't write any application. SIM Toolkit Application development platform also very limited to few circle of developers, while the hardware platform (cellphone with SIM Toolkit enabled) adopted at an exponentially rate since 2001.

Another type of smart card platform is JavaCard, it does offered powerful tools for building smart card application and with support of it's more than 1900 developers (since 1996) in their community, JavaCard later adopted to become the de-jure standard for building EMV application (application for authenticating transaction on chip based credit/banking card).

SIM Toolkit is an interesting protocol; it offered mechanism to control the transaction protocol directly from card application, unlike a normal card application (Java Card for example). By enabling transaction protocol controlled directly from the card, it enabled the application executed on different type of card terminal (a cellphone is an example of terminal in-case for SIM toolkit, it enabled different type of MNO issued application to run on top of it, an example of popular SIM toolkit application would be M-PESA which is issued by Vodafone)

Orb-Weaver card is platform that enable developer to build SIM toolkit application on top of it to run on any Orb-Weaver enabled terminal, with better scalability to interface 3rd party device or application (added to the protocol) and extra security such as secure channel from Global Platform (which can be activated/deactivated by application). Both ETSI/3GPP specs of SIM Toolkit protocol are implemented at framework level while system APIs provides the input-output interface to Orb-Weaver enabled terminal.

Smart Card Terminal Platform

Most of conventional smart card terminal (example EFTPOS terminal) employs native application development on top of Linux/BSD (UNIX derivative) operating system, newer type of smart card terminal already using Android operating system capable running any Android application. Use of mobile operating system instead of embedded operating system require significant hardware resources such as processor and memory, a typical 1GHz processor (ARM Cortex A7, standard application processor) and a minimum of 32MB RAM (the minimum DDR chip memory size) which increase the production cost for each terminal.

Introducing a Leaf Operating System for ARM Cortex M Series and less than quarter megabyte for RAM size (that's order of magnitude reduction in resource usage), it offered the minimum terminal functionality for interfacing both contactless and contact card, also equipped with Orb-Weaver virtual machine for executing user application written in Orb-Script language.

The programming paradigm for both terminal and smart card device is the same, which means the same program can run on both the terminal and the card device (strictly limited to use only common framework) resulting in the same application behavior.

Web Client-Server Scripting Platform (WCS)

Orb-Weaver is designed to manipulate/process data and control program flow, a generic features for server side scripting, most of current server side scripting rely on PHP, which is actually a string processor software, web content are dynamically generated by PHP script on web server and loaded onto web browser, a script (typically JavaScript) inserted into to content on the web browser to perform additional task to web server. Most of current web development platform

separate both web client and web server programming into different platform, some combine PHP and JavaScript, others might use JavaScript and Ruby. Through an Orb-Weaver framework, developer could automate the content of script generated on web client and control which type request (additional task) that will be sent into web server (AJAX) and trigger a specific event through a specialized API protocol (OW-Stream which based on REST protocol).

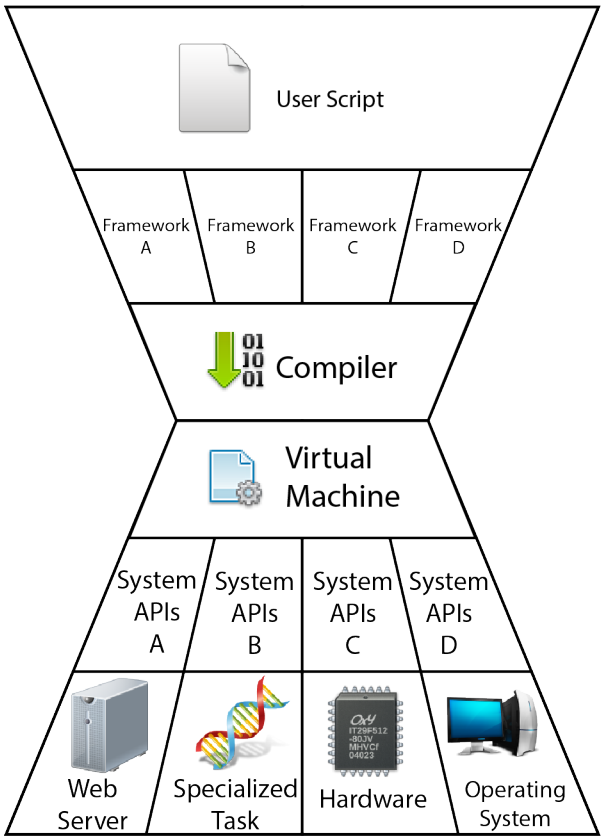
Specialized Platform

Orb-Weaver can be used as platform to perform specialized task through the use of it's specialized system APIs, in-order to advance our Orb-Weaver platform to different type of task, one way is to open-source the APIs sourcecode, static library or dynamic libarary which can be loaded upon run-time, therefore different type of developer from different background could integrate their APIs within Orb-Weaver.

Orb-Script

Is high level, multi-paradigm, multi-platform programming language, it is designed for easy migration from JavaScript environment, so it support most JavaScript syntax (we're not advocating people to use JavaScript, but rather as slide-in process during migration to achieve inverse tangent learning curve), but unlike JavaScript it offered several features that mostly available in commercial language such as macro, encapsulation, system APIs, events, it also provide semi-object oriented programming at loosely coupled environment. The purpose of Orb-Script is to provide high level language abstraction for programmers to interface with the Orb-Weaver virtual machine, this done either through system APIs or framework.

For scalability reason Orb-Script supports sharing of user classes through framework, it also acts as 3rd party driver to interface with different hardware/software platform, we called this concept **Hourglass**.



through hourglass concept, user script can be re-target to different hardware platform by only changing it's framework, a set of framework defined for this purpose called common framework. Common framework consist of several basic APIs such as standard input-output and other function most commonly used during programming, another framework called Specialized framework contain hardware/platform specific function, executing this function on another platform might

result in unpredictable behavior, therefore a measurement taken by encapsulating it's function within specialized framework which will be handled by virtual machine.

Lazy Language

Lazy language for lazy programmer, Orb-Script designed to be lazy language like it's predecessor, but based on blub paradox, in-order for Orb-Script to become a powerful language it need to evolve and increase it's expression support, lazy language provide the necessary bounds of how an Orb-Script will be.

Lambda Expression

Orb-Script support lambda expression, lambda is a type of code sequence that can be pass into different function to be executed, this kind of function called high order function, this code sequence later can be registered as event handler or executed by another thread concurrently.

Object and Array

Orb-Script supporting object and array sub-consciously, and object or array are represented in ASN/1 notation and can be passed either as argument or return value to another function. Object or array can be modified during runtime or converted to JSON equivalent.

Context Oriented Programming

Context oriented programming is a type of programming paradigm that predates Object Oriented Programming, context are generated by system APIs the similar way with file interface on C and PHP, it provide the standard interface for input-output (read, write, seek, close), the actual execution later depends on system APIs that provide the context, it can be used to interface another 3rd party device, socket, file, session, etc.

Object Oriented Programming

Orb-Script also provide semi-Object Oriented Programming paradigm on loosely coupled environment, it support the necessary instance creation and encapsulation for user defined class, but it doesn't support inheritance, as inheritance will only increase memory usage during runtime.

Mathematical Axiom

Orb-Script didn't provide advanced mathematical axiom, as it didn't intended for complex mathematical calculation, the maximum data size of mathematical operation are still within integer value. The decision not to include advanced algebraic data types and operation is to minimized the performance issue on constrained environment and reduce the effort taken for porting purpose, advanced mathematical axiom can be implemented as system APIs (see Data-Type Extension).

String and Bytes

Orb-Script was first designed to manipulate string or bytes back in 2014, several features such lambda expression, object/array management, context oriented programming and object oriented programming later appeared during development process

Data-Type Extension

Data-type extension enabling custom mathematical operation to be performed at variable basis, custom variable are assigned by system API along with it's custom mathematical operation, it enable advanced mathematical operation to be performed as simple as adding/subtracting/multiplying/dividing a variable.