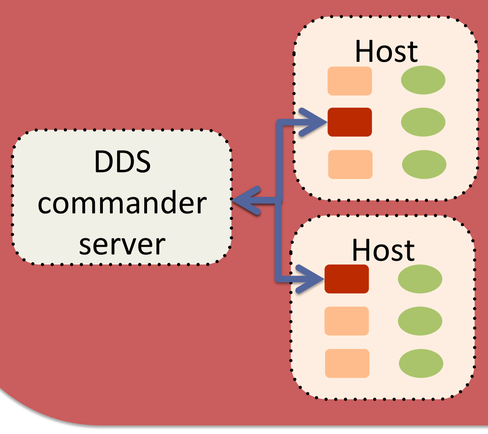
Dynamic Deployment System

Andrey Lebedev and Anar Manafov
GSI, Darmstadt, Germany

Motivation

Create a system, which is able to **spawn and control** hundreds of thousands of different user's tasks which are tied together by a **topology**, can be run on **online clusters** or **computing clusters** which use **different resource management systems (RMS)** or even on a laptop and can be **controlled by external tools**.



DDS is being developed in frames of **ALFA** (**ALICE-FAIR**) project.

Current release
v2.0

Basic concepts

Single responsibility principle command line **tool-set and API**; users' task is a black box – it can be an **executable or a script**; **watchdogging; rule-based execution of tasks**; **plug-in system** to abstract from RMS including **SSH** and a **localhost** plug-ins; **doesn't require pre-installation and pre-configuration** on the worker nodes; private facilities on demand with **isolated sandboxes**; key-value propagation and messaging.

DDS core implements an event-driven async architecture.

DDS

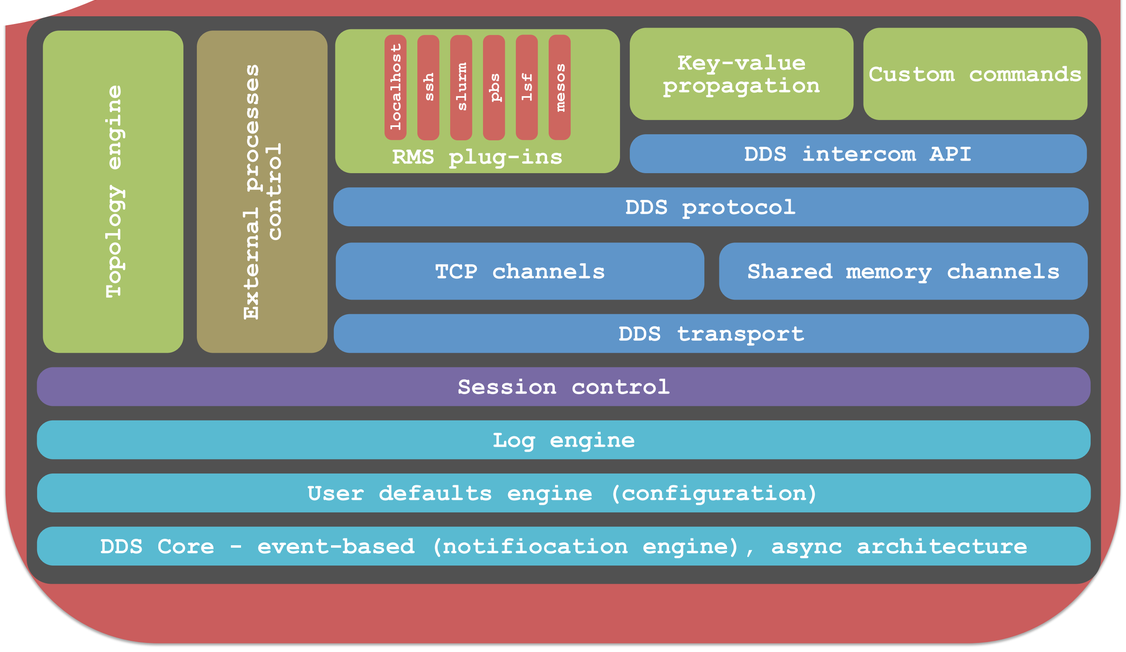
Property propagation

feature allows user's tasks to **exchange and synchronize the configuration (key-value)** dynamically at runtime. For example, in order to **synchronize the startup** of the user's tasks. It is **highly optimized** for massive key-value transport and has a **decentralized architecture**.

Custom messaging

feature allows **user's tasks** and **ext. utilities** to **exchange messages**, so-called, custom commands. It can be used, for example, as a basis for a **control system**.

10000 foot view



From user's perspective

Topology

```
<topology id="myTopology">
   [... Definition of tasks,
properties, and collections ...]
   <main name="main">
      [… Definition of the topology
itself, including ...]
   </main>
</topology>
```

CLI tools

```
dds-session
dds-agent-cmd
dds-custom-cmd
dds-info
dds-prep-worker
dds-server
dds-stat
dds-submit
dds-test
dds-topology
dds-user-defaults
```

Intercom API

```
CIntercomService service;
CKeyValue keyValue(service);

// Subscribe on key update events
keyValue.subscribe([](
        const string& _propertyID,
        const string& _key,
        const string& _value)
{...});

// Start listening to events we have
subscribed on
service.start();
```

More info
http://dds.gsi.de

DDS@GitHub
https://github.com/FairRootGroup/DDS