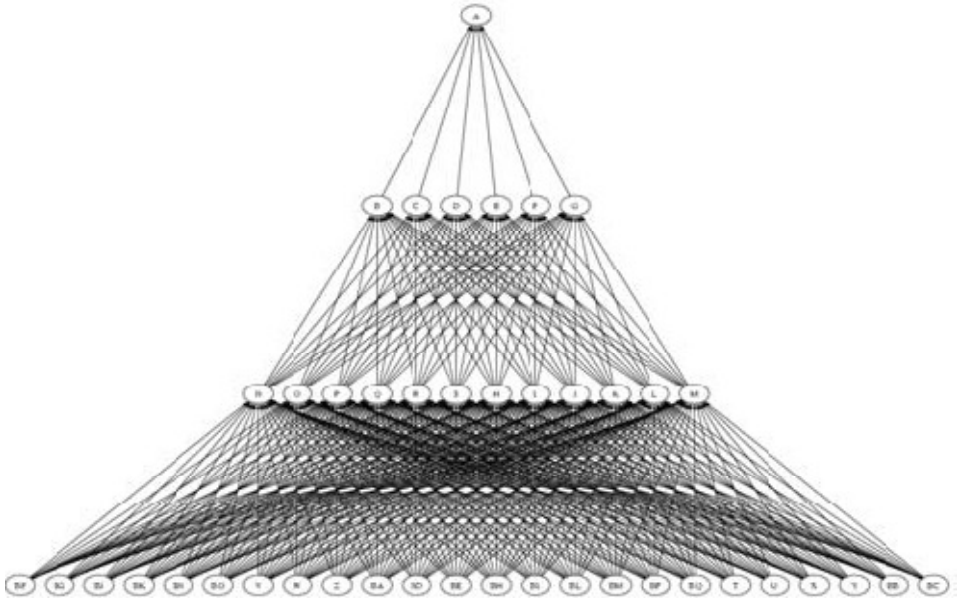
Grid Application Description Languages



Picture taken from http://www.globus.org/alliance/publications/papers/VDS02.pdf
Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation

Application Description Languages

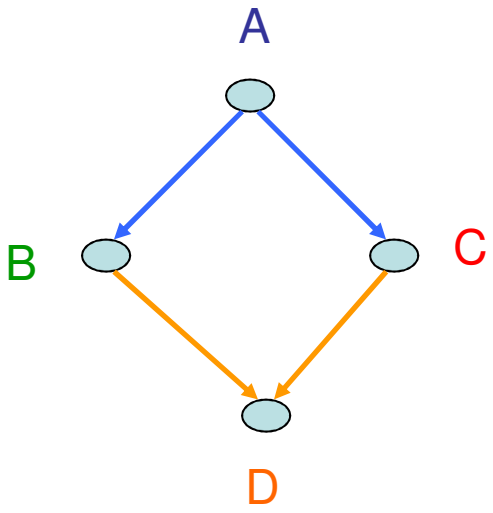
- Allow the user to specify characteristics of their applications to be run on the grid
- Script-like
- Workflow-based
- Most of them assume that the application can be represented as a set of jobs
  - DAG where nodes represent jobs and edges represent job precedence

Application Description Languages

- As such, traditional clustering/scheduling techniques can be applied to map graph nodes to grid nodes
- Some grid application description languages:
  - VDL
  - Condor DAGMan
  - JSDL
  - GXML
  - AGWL
  - XPWSL
  - GEL
  - GRID-ADL
  - JDL
  - .....

Languages: DAGMan

```
#
# first_example.dag
#
Job A A.condor
Job B B.condor
Job C C.condor
Job D D.condor
PARENT A CHILD B C
PARENT B C CHILD D
```



Languages: DAGMan

```
#
# task A
#
executable = A.exe
input = test.data
output = A.out
log = dag.log
Queue
```

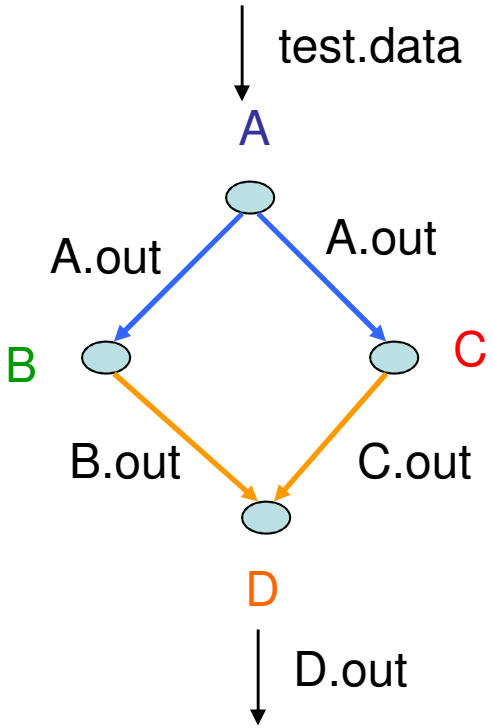
```
#
# task B
#
executable = B.exe
input = A.out
output = B.out
log = dag.log
Queue
```

```
#
# task C
#
executable = C.exe
input = A.out
output = C.out
log = dag.log
Queue
```

```
#
# task D
#
executable = D.exe
input = B.out C.out
output = final.out
log = dag.log
Queue
```

Languages: VDL

```
TR calculate{ output b, input a} {
        app vanilla = “calculate.exe";
        arg stdin = ${output:a};
        arg stdout = ${output:b};
}
TR analyze{ input a[], output c} {
        app vanilla = "analyze.exe";
        arg files = ${:a};
        arg stdout = ${output:c};
}
DV calculate { b=@{output:A.out},
               a=@{input:test.data} };
DV calculate { b=@{output:B.out},
               a=@{input:A.out} };
DV calculate { b=@{output:C.out},
               a=@{input:A.out} };
DV analyze{ a=[ @{input:B.out},
                @{input:C.out} ],
            c=@{output:D.out} };
```



VDL

- Graph generated by Pegasus (Planning for Execution in Grids) through the analysis of the TRs and DVs
- Control is given to Condor DAGMan, once the graph is generated

Languages: GXML

Part of the GANGA framework

(Grid Application iNformation Gathering and Accessing)

```
<flow:Workflow flow:start="A">
    <jsdl:Job jsdl:id="A">
    ...
    </jsdl:Job>
    <jsdl:Job jsdl:id="B">
    <flow:Depend flow:success="A"/>
    ...
    </jsdl:Job>
    <jsdl:Job jsdl:id="C">
    <flow:Depend flow:success="A"/>
    ...
    </jsdl:Job>
    <jsdl:Job jsdl:id="D">
    <flow:Depend flow:success="B & C"/>
    ...
    </jsdl:Job>
</flow:Workflow>
```

flow:LoopCount tag allows loops

Converted to a DAG and managed by Condor DAGMan

Languages: AGWL
(Abstract Grid Workflow Language)

```
<agwl-workflow>
    <activity name="A" type="teste:A">
        <dataIn name="test.dat" > </dataIn>
        ...
        <dataOut name="A.out"> </dataOut>
    </activity>
    <subWorkflow name="tasksBandC">    // defining tasks B and C
    <body>
        <activity name="B" type="teste:B">
            <dataIn name="A.out" > </dataIn>
            ...
            <dataOut name="B.out"> </dataOut>
        </activity>
        <activity name="C" type="teste:C">
            <dataIn name="A.out" > </dataIn>
            ...
            <dataOut name="C.out"> </dataOut>
        </activity>
    </body>
    </subWorkflow>
    <activity name="D" type="teste:D">
        <dataIn name="B.out" > </dataIn>
        <dataIn name="C.out" > </dataIn>
        ...
        <dataOut name="D.out"> </dataOut>
    </activity>
</agwl-workflow>
```

Converted to CGWL
(Concrete Grid...)
and executed by ASKALON

Languages: XPWSL

XML-based Parallel Workflow Specification Language

```
<header>
    <name>DAG example</name>
    <description>same example previously
    used</description>
</header>

<assignment>
    <task id="T0" delay="delay_A">
            <path>/path</path>
            <code>A.exe</code>
    </task>
    <task id="T1" delay="delay_B">
            <path>/path</path>
            <code>B.exe</code>
    </task>
    <task id="T2" delay="delay_C">
            <path>/path</path>
            <code>C.exe</code>
    </task>
    <task id="T3" delay="delay_D">
            <path>/path</path>
            <code>D.exe</code>
    </task>
</assignment>
```

```
<datalink>
    <block type="sequential">
        <task_id>T0<task_id>
        <multi>1</multi>
        <input>test.dat</input>
    <block>
    <block type="sequential">
        <task_id>T1<task_id>
        <multi>1</multi>
        <input>A.out</input>
        <task_id>T2<task_id>
        <multi>1</multi>
        <input>A.out</input>
    <block>
    <block type="sequential">
        <task_id>T3<task_id>
        <multi>1</multi>
        <input>B.dat</input>
        <input>C.dat</input>
    <block>
</datalink>
```

Languages: GEL
Grid Execution Language

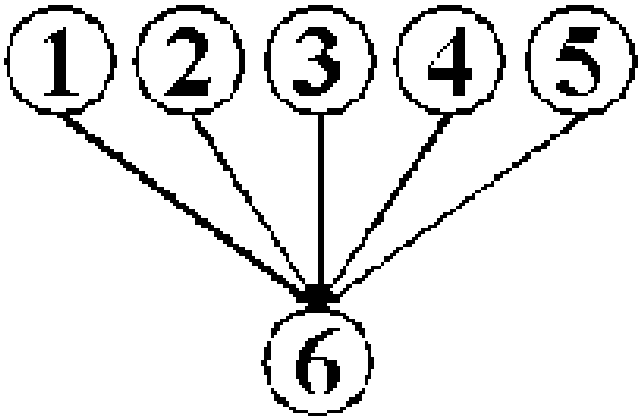
```
taskA = {exec="A.exe"; dir="/path"; args="test.dat"}
taskB = {exec="B.exe"; dir="/path"; args="A.out"}
taskC = {exec="C.exe"; dir="/path"; args="A.out"}
taskD = {exec="D.exe"; dir="/path"; args="B.out","C.out"}
taskA; taskB | taskC; taskD
```

Languages: GRID-ADL

Grid Application Description Language

```
graph loosely-coupled
task A -e A.sub -i data.in -o a.out
task B -e B.sub -i a.out -o b.out
task C -e C.sub -i a.out -o c.out
task D -e D.sub -i b.out c.out -o data.out
```

GRID-ADL




```
graph phase
OUTPUT = ""
foreach ${TASK} in 1..5 {
  task ${TASK} -e ${TASK}".exe"
       -i ${TASK}".in" -o ${TASK}".out"
  OUTPUT = ${OUTPUT} + ${TASK}+".out  "
}
task 6 -e 6.exe -i ${OUTPUT} -o data.out
transient ${OUTPUT}
```

Languages

Lang	Yr	Mw	RMS	Type	Wflw	DAG	DAGInf
DAGMan	02	DM	C	plain	no	yes	manual
VDL	02	DM	C	plain	yes	yes	auto
GXML	05	G	G / C	XML	yes	yes	manual
AGWL	05	AS	Glob	XML	yes	yes	manual
XPWSL	05	Join	Join	XML	yes	yes	manual
GEL	05	Gel	Vars	script	no	yes	auto
GRID-ADL	04	AM	AM/ PBS	script	no	yes	auto

G: Ganesh DM: DAGMan AS: Askalon AM: AppMan Vars: SMP, SGE, LSF, PBS, or Globus