CS528

Budget Aware Scheduling
&
Reliability and Robustness

A Sahu

Dept of CSE, IIT Guwahati

Outline

- Scheduling Workflow in Cloud DC
  - Budget Aware
  - End-End Delay Minimization
- Reliability
  - Workflow scheduling
  - Backup copy

Reliability and Robustness

- *Robustness* is
  - a quality defined by comparison
  - the ability to continue to function beyond the period committed until it wears out.
  - The ability of software systems to react appropriately to abnormal conditions.
- Robust means stronger than *Reliable*
- *Reliability* is
  - ***numerically defined as a probability*** of success over time at given conditions.
  - Trust worthy
  - A concern encompassing correctness and robustness.

Reference:

Wu et.al, *End to End Delay Minimization for Scientific Workflow in cloud under Budget Constraints,* IEEE Trans. On Cloud Computing. 2015.

Introduction

- With emergence of cloud computing and rapid deployment of cloud infrastructures
  - Number of **scientific workflows** have been shifted to cloud environments.
- Challenges:
  - Reducing financial cost in addition
  - to meeting the traditional goal : performance
- Quick evaluation of scientific workflow
  - to minimize the workflow end-to-end delay under a user-specified financial constraint

Scientific Workflow : SWF

- Large-scale scientific computing tasks
  - for data generation, processing, and analysis are
  - often assembled and constructed as **Workflows**
  - comprised of many interdependent **modules**
- Workflow (WF) module communicates
  - with others through the sharing of data sets,
  - which are either stored in shared file system or
  - transferred from node to node by WF management system
- Scientific Workflows are
  - typically executed in a distributed manner
  - in heterogeneous network environments

WF in Clouds System

- It is essential construct analytical models
  - to quantify the network performance of scientific workflows
  - in IaaS cloud environments,
  - and formulate a task scheduling problem
- Scheduling Problems: WF on Cloud
  - to minimize the workflow end-to-end delay
  - under a user-specified financial cost constraint,
- Referred to as Minimum End-to-end Delay under Cost Constraint (MED-CC)

Workflow Execution in Cloud

- Workflow is represented as
  - a directed acyclic graph (DAG),
- Submitted to the workflow engine
  - for executing, scheduling, tracking and reporting
- Workflow (WF) have independent tasks (Work/W), and
  - Execution model with inter-module dependencies
  - Identify and quantify the key financial and time cost

Cost Model : time and financial

- **Time Cost or simply Time :** overall time to execute Work W_i on VM_j

$$T_{i,j}=T(I_j)+T(E_{i,j})+T(R_i)$$

  - $T_{i,j}$= overall time to execute Work W_i on VM_j
  - $T(I_j)$ = Startup Time for VM_j
  - $T(E_{i,j})$ = time to execute W_i on VM_j
  - $T(R_i)$ = time of upload/download data from/to VM_j

Cost Model : time and financial

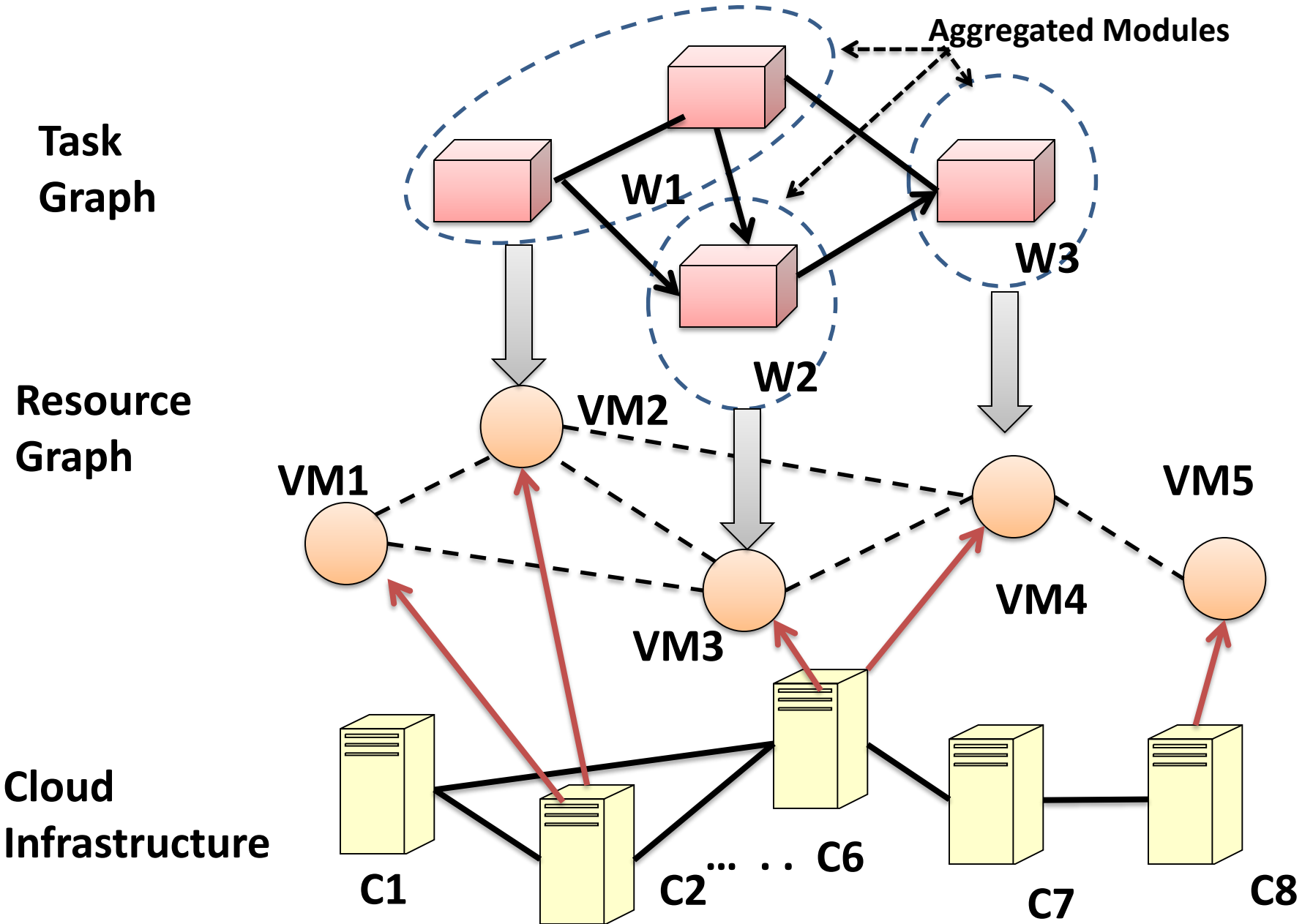
- **Financial Cost (or simply Cost): O**verall time to execute Work W_i on VM_j

$$C_{i,j}=C(I_j)+C(E_{i,j})+C(R_i)+C(S_i)$$

  - $C(S_i)$ = data storage cost of W_i
  - **$C(I_j)$, $C(E_{i,j})$, $C(R_i)$** cost of Init VMj, execution of Wi on VMj and download/upload data to/from VM_j for W_i
- Set of Available VM type $VT=\{vt_0,vt_1,..vt_{n-1}\}$
  - **vt_j have processing power p_j and cost c_j**
- Cost of executing W_i on VM_j

$$C(E_{i,j})=T(E_{i,j})*c_j$$

Modeling Workflow Exe in Cloud



3 Layer Models of WF Exe^{n} in Cloud

- Three layers in the work-flow scheduling problem in cloud environments
  - **Task graph layer** comprised of interdependent workflow modules,
  - **Resource graph layer** representing a network of VMs,
  - **Cloud infrastructure layer** consisting of physical computer nodes connected by network links

Work mapping to VM

- **Task graph layer** : scientific work- flows
  - preprocessed by an appropriate clustering technique
  - based on the inter-module dependencies
  - and the volumes of inter-module data transfer
- A group of modules in the original workflow are
  - bundled together as one aggregate module in the resulted task graph
- Simple mapping: one-to-one mapping scheme
  - **Each aggregate module** in the task graph is
  - **Assigned to a different VM** for execution.

Layer1 : Workflow Model

- Task graph/workflow modeled as DAG

$$G_w(V_w, E_w)$$

  - Consisting of V_w modules
  - with E_w directed edges $l_{i,j}$ representing the data dependency between module w_i and w_j
  - Each module $w_i \epsilon V_w$ carries a certain amount of workload WL_i
  - Each edge $l_{i,j}$ transfers a certain data size $DS_{i,j}$.

VM Cost and Perf. Model

- A set of available VM types

$$VT = \{ vt_0, vt_1, \ldots vt_n \}$$

  - where each type vt_j is associated with both cost- and performance-related attributes vt_j as : $vt_j (vp_j, Cv_j)$

- vp_j : Processing power of VM type vt_j
  - including its processor, speed, disk volume and memory space
- Cv_j : Financial cost for using this VM
  - Per time unit including all cost components such as
  - VM initialization, module execution, and data transfer

Layer3: Cloud Infrastructure Model

- cloud infrastructure is model as arbitrary weighted graph $G_c=\{V_c, E_c\}$
  - consisting of a set of physical computer nodes.
  - Each computer node $c_i \in V_c$ has a processing power PP_i
- Cost of a data transfer R_{ij} of size DS_{ij} from module w_i to w_j is calculated as:

$$C(R_{i,j}) = C_R \cdot DS_{i,j}$$

Layer 2 :Virtual Resource Model

- Virtual Resource : Modeled as fully connected weighted graph $G'_c(V'_c, E'_c)$
  - consisting of $|V'_c|$ VMs connected by $|E'_c|$ virtual links.
  - Each virtual link $l'_{p,q}$ has a bandwidth $BW'_{p,q}$,
  - which is a function of physical bandwidth between two physical machines provisioning VMs c'_p and c'_q.
  - Each VM $c' \in V'_c$ is an instance of a certain VM type.
- Time of data transfer R_{ij} of size DS_{ij} from module w_i to w_j is calculated as

$$T(R_{i,j}) = DS_{i,j}/BW'_{pq} + d'_{pq}$$

  - where d'_{pq} is delay of the virtual link

Time and Cost Execution W_i on VM_J

- Time of Execution E_{ij} of module w_i on a VM of type vt_j

$$T(E_{ij}) = Wl_i / vp_j$$

  - Wl_i is workload of w_i and
  - vp_j is performance of vt_j of VM_j
- Module execution cost is calculated as:

$$C(E_{ij}) = T(E_{ij})\, C_{Vj}$$

Min End2End Exe Cost Constraints

- MED-CC Problem : Given DAG $G_w(V_w, E_w)$, VT set $VT = \{vt_0, vt_1, \ldots vt_{n}\}$ and budget **B**
- Find a Schedule S, such that
  - minimum end-to-end delay of the one-to-one mapped workflow is achieved

MED= $\min_{all\ possible\ S}(T_{total}) = \min_{all\ possible\ S}\left(\sum_{all\ w_i \in CP} T_{i,j}\right)$

- Subjected Cost Constraints $C_{total} = \sum_{i=0}^{m-1} C_{i,j} \leq B$
- Proved to be NPC
  - Using Multiple Choice Knapsack Problem (MCKP)

Critical Greedy Heuristics

CriticalGreedy(G, VT, B){

1. For all w_i, for all vt_j calculate C_{ij} and T_{ij}
2. Find Minimal Cost C_{min}
3. if (B < C_{min}) **No Solution, return;**
4. Find Maximal Cost C_{max},
5. if (B>C_{max}) **Map all w_i to best vt_{max} return;**
6. ***AllcateVM using global budget level();***
7. ***Do local adjustment ();***

}

Allocate VM using Global Budget Level

Try to allocate good VT based on Budget

$$GBL = (B-C_{min})/(C_{max}-C_{min})$$

For all $\mathbf{w_i}$ map to $\mathbf{vt_j}$ with GBL

Target cost for $\mathbf{w_i}$, $\mathbf{tc_i = C_{i,min}+(C_{i,max}-C_{i,min})*GBL}$

Target VT for $\mathbf{w_i}$, for $\mathbf{min(tc_i-C_{i,k})}$ and $\mathbf{C_{i,k}<tc_i}$

Do local adjustment ();

While **cost2Spend > 0** {

Calculate Current Critical path CP;

For all w_i in ***CP*** for all vt_j

find max benefit with $\Delta T_{i,j} > 0$ and **max** $\frac{\Delta T_{i,j}}{\Delta C_{i,j}}$

//Putting small money but max time reduction

if such w_i found change mapping of w_i to new vt_j

cost2Spend -= ExtraCostSpendonThis;

}

Reference:

Poola et.al, ***Robust Scheduling of Scientific Workflows with Deadline and Budget Constraints in Clouds,*** IEEE Int. Conf. on Adv. Info. Networking & Applications,. 2014.

Introduction

- **Scientific workflows** Scheduling in Cloud
  - Cost and Deadline common criteria
- Reliability and Robustness is also Important
- Robust scheduling
  - that handles performance variations of Cloud resources and
  - failures in the environment is essential in the context of Clouds
- Robust and fault-tolerant schedule
  - while minimizing makespan.

Failure in Cloud

- Failures also affect the overall workflow execution and increase the makespan.
- Failures in a workflow application are types
  - Task failures, VM failures, WF level failures
- Task failures may occur due to
  - dynamic execution environment configurations,
  - missing input data, or
  - system errors.
- VM failures are caused by
  - hardware failures and load in the datacenter

Failure in Cloud

- Workflow level failures can occur due to
  - server failures, Cloud outages,
- Prominent fault tolerant techniques that handle such failures are
  - retry, alternate resource, check-pointing, and replication
- Workflow management systems
  - should handle performance variations and
  - failures while scheduling workflows

Robust Scheduler

- A schedule is said to be robust if
  - it is able to absorb some degree of uncertainty
  - in the task execution time
- Robust schedules are much needed in
  - mission-critical applications and
  - time-critical applications
- Robust and fault-tolerant scheduling algorithms
  - identify these aspects and provide a schedule
  - that is insensitive to these uncertainties
  - by tolerating variations and failures
  - in the environment up to a certain degree.

Robust Scheduler

- Robustness of a schedule is always
  - measured with respect to another parameter such as makespan, schedule length
- Robustness is usually achieved
  - with redundancy in **time or space**
  - Adding **slack time** or **replication of nodes**.
- Robust Scheduling Approach
  - efficiently maps tasks on resources
  - judiciously adds slack time based on the deadline and budget constraints