مدل های برنامه نویسی و الگوهای طراحی

دکتر محمد کاظم اکبری
مرتضی سرگلزایی جوان

http://crc.aut.ac.ir
سرفصل مطالب

• انواع بار کاری
• انواع کاربردها
• بررسی چند سناریو
• مکانیزم ها و الگوهای توسعه
انواع بار کاری

• Static
• Periodic
• Once-in-a-lifetime
• Unpredictable
• Continuously Changing
IT resources with an equal utilization over time experience Static Workload.
Periodic Workload

IT resources with a peaking utilization at reoccurring time intervals experience periodic workload.
Once-in-a-lifetime Workload

IT resources with an equal utilization over time disturbed by a strong peak occurring only once experience Once-in-a-lifetime Workload.
Unpredictable Workload

IT resources with a random and unforeseeable utilization over time experience unpredictable workload.
IT resources with a utilization that grows or shrinks constantly over time experience Continuously Changing Workload.
Social Networks, Enterprise Applications

Frameworks for Cloud Application Development

Virtual hardware, networking, OS images, and storage.

Applications (SaaS)

Middleware (PaaS)

Hardware and OS (IaaS)
انواع کاربردهای ابری

- Pipelines and Batch processing systems
- HPC and Scientific Computing
- Web and Mobile applications
- Distributed File Systems and Storage
- Gaming
- IoT, IoS and Industry 4.0 Systems
Pipelines and Batch Processing

• **Indexing.** The processing pipeline supports indexing of large datasets created by Web crawler engines.

• **Data mining.** The processing pipeline supports searching very large collections of records to locate items of interests.

• **Image processing.** A number of companies allow users to store their images on the cloud (e.g., Flickr (www.flickr.com) and Google (http://picasa.google.com/)). The image-processing pipelines support image conversion (e.g., enlarging an image or creating thumbnails). They can also be used to compress or encrypt images.

• **Video transcoding.** The processing pipeline transcodes from one video format to another (e.g., from AVI to MPEG).

• **Document processing.** The processing pipeline converts very large collections of documents from one format to another (e.g., from Word to PDF), or encrypts the documents. It could also use optical character recognition (OCR) to produce digital images of documents.
Pipelines and Batch Processing

• Generation of daily, weekly, monthly, and annual activity reports for organizations in retail, manufacturing, and other economic sectors.

• Processing, aggregation, and summaries of daily transactions for financial institutions, insurance companies, and healthcare organizations.

• Inventory management for large corporations.

• Processing billing and payroll records.

• Management of the software development (e.g., nightly updates of software repositories).

• Automatic testing and verification of software and hardware systems.
Scheduler (Task orchestrator)

Datasets

Tasks

Scheduler

Elastic Sizing (EC2, ...,
Scalability engine

Amazon S3: Video storage

EC2 instances: video rendering

Amazon SQS: Web front end

Animoto

RightScale Cloud Management Platform
Scheduler

Networked LAN

Maya: Train Design

MayaRender

Render Files

Scenarios

Render Tasks

Scheduler

Networked LAN
Programming Models

- Task Programming Model
- Thread Programming Model
- Map Reduce Programming Model
- Parameter Sweeping Model
- .. Implement your own..
Task Programming Model

• Used to model Independent **Bag of Tasks** (BoT) applications
  • The application is a *collection of execution unit*
  • Each execution unit is not related to the others
  • There is **no order in the execution** of the units

```csharp
public DumbTask: ITask
{
    ...
    public void Execute()
    {
        ......
    }
}

for(int i=0; i<n; i++)
{
    ...
    DumbTask task = new DumbTask();
    app.SubmitExecution(task);
}
```
Thread Programming Model

- Based on the concept of distributed thread
  - Like a local thread but executed remotely
  - Implements a subset of the common operations on thread
    - Start
    - Stop
    - State Query
    - Join
- Provides a quick way for porting on a distributed middleware, multi-threaded applications
Parameter Sweeping Model

• Based on the Task Programming Model
• Provides a set of facilities to run applications where
  • ..all the tasks are homogeneous (same task)
  • ..the specific instance of tasks is specialized by parameters
  • ..all the possible combination of parameters are explored by generating a task instance for each of the combinations
Map Reduce Programming Model

The diagram illustrates the Map Reduce programming model, showing the flow from an application to a master instance, then to various map and reduce instances, each handling data segments and local disks. The process involves the map phase where data is distributed and processed, followed by the reduce phase where results are aggregated.
Map Reduce in the Cloud
## HPC and Scientific Computing

<table>
<thead>
<tr>
<th>System</th>
<th>DGEMM Gflops</th>
<th>STREAM GB/s</th>
<th>Latency μs</th>
<th>Bndw GB/S</th>
<th>HPL Tflops</th>
<th>FFTE Gflops</th>
<th>PTRANS GB/s</th>
<th>RandAcc GUP/s</th>
</tr>
</thead>
<tbody>
<tr>
<td>Carver</td>
<td>10.2</td>
<td>4.4</td>
<td>2.1</td>
<td>3.4</td>
<td>0.56</td>
<td>21.99</td>
<td>9.35</td>
<td>0.044</td>
</tr>
<tr>
<td>Franklin</td>
<td>8.4</td>
<td>2.3</td>
<td>7.8</td>
<td>1.6</td>
<td>0.47</td>
<td>14.24</td>
<td>2.63</td>
<td>0.061</td>
</tr>
<tr>
<td>Lawrencium</td>
<td>9.6</td>
<td>0.7</td>
<td>4.1</td>
<td>1.2</td>
<td>0.46</td>
<td>9.12</td>
<td>1.34</td>
<td>0.013</td>
</tr>
<tr>
<td>EC2</td>
<td>4.6</td>
<td>1.7</td>
<td>145</td>
<td>0.06</td>
<td>0.07</td>
<td>1.09</td>
<td>0.29</td>
<td>0.004</td>
</tr>
</tbody>
</table>

Parallel Programming Environments

Compilers, PVM, MPI, ….

Cluster Middleware:
Single System Image and Availability Infrastructure

PC / Workstation
Common SW & Applications
Operating System
Network Interface HW

PC / Workstation
Common SW & Applications
Operating System
Network Interface HW

PC / Workstation
Common SW & Applications
Operating System
Network Interface HW

PC / Workstation
Common SW & Applications
Operating System
Network Interface HW

High speed network connection
AWS cfncluster

cfncluster is a framework that deploys and maintains HPC clusters on AWS
Web and Mobile applications

• Applications built upon cloud platforms (e.g. Salesforce, Bluemix)
• Periodic or a temporary presence (Web sites for conferences or other events or the holiday seasons)
• Mashups - an application that uses and combines data, presentation, or functionality from two or more sources to create a service.
• Reduce Lack of resources at Mobile devices
• Office and Business Application
Force.com Platform

Metadata Cache

Bulk Processing Engine

Multitenant-aware Query Optimiser

Runtime Application Generator

Full-text Search Engine

Indexes

Virtual Application Components

- COMMON APPLICATION SCREENS
- TENANT-SPECIFIC SCREENS
- OBJECTS (TABLES)

Shared Database

- Data Tables
- Pivot Tables
- Metadata Tables (UDD)

User-base 1: Application 1

User-base 2: Application 2

User-base 3: Application 3
Storage

• Backup & Archive
• Sharing
• Collaboration
• Synchronization
• Distribution
Amazon S3 and Glacier
Gaming

- **video streaming**
  - direct and on-demand streaming of games onto computers, consoles and mobile devices, similar to video on demand, through the use of a thin client.
  - The actual game is stored, executed, and rendered on the remote operator's or game company's server and the video results are streamed directly to a consumer's computers over the internet.

- **file streaming**
  - actual game is run on the user's gaming device such as a mobile device, a PC or a console.
  - A small part of a game, usually less than 5% of the total game size, is downloaded initially so that the gamer can start playing quickly.
  - The remaining game content is downloaded to the end user's device while playing.
AWS Case Study: Supercell

- Finland-based Supercell, founded in 2010 by 6 game industry veterans, is one of the fastest-growing social game developers in the world. With a staff of just over 100 employees, Supercell’s three games—Hay Day, Clash of Clans, and Boom Beach—attract tens of millions of players on iOS and Android devices every day.
AWS AppStream
IoT, IoS and Industry 4.0 systems

- Smart Factory
- Smart home / building
- Smart city
- Smart agriculture
- Smart logistics
- Smart energy
- Smart retail
- …

<table>
<thead>
<tr>
<th>Reason</th>
<th>Respondents Who Agree</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improved system reliability and availability</td>
<td>50%</td>
</tr>
<tr>
<td>Pay only for what you use</td>
<td>50%</td>
</tr>
<tr>
<td>Hardware savings</td>
<td>47%</td>
</tr>
<tr>
<td>Software license savings</td>
<td>46%</td>
</tr>
<tr>
<td>Lower labor costs</td>
<td>44%</td>
</tr>
<tr>
<td>Lower maintenance costs</td>
<td>42%</td>
</tr>
<tr>
<td>Reduced IT support needs</td>
<td>40%</td>
</tr>
<tr>
<td>Ability to take advantage of the latest functionality</td>
<td>40%</td>
</tr>
<tr>
<td>Less pressure on internal resources</td>
<td>39%</td>
</tr>
<tr>
<td>Solve problems related to updating/upgrading</td>
<td>39%</td>
</tr>
<tr>
<td>Rapid deployment</td>
<td>39%</td>
</tr>
<tr>
<td>Ability to scale up resources to meet needs</td>
<td>39%</td>
</tr>
<tr>
<td>Ability to focus on core competencies</td>
<td>38%</td>
</tr>
<tr>
<td>Take advantage of the improved economies of scale</td>
<td>37%</td>
</tr>
<tr>
<td>Reduced infrastructure management needs</td>
<td>37%</td>
</tr>
<tr>
<td>Lower energy costs</td>
<td>29%</td>
</tr>
<tr>
<td>Reduced space requirements</td>
<td>26%</td>
</tr>
<tr>
<td>Create new revenue streams</td>
<td>23%</td>
</tr>
</tbody>
</table>
Integration!
سطح استفاده از رایانش ابری

- Service Level
- Runtime Level
- Container (Microservice) Level
- Virtual Machine Level
- Bare metal Level
(GeoCoder) API / SDK مثالی از

```
$ pip install pygeocoder

#!/usr/bin/env python3
from pygeocoder import Geocoder
if __name__ == '__main__':
    address = 'Amirkabir University of Technology, Hafez, Tehran'
    print(Geocoder.geocode(address)[0].coordinates)

$ python3 search1.py
(35.7042932, 51.4096345)
```
(ideone.com) Runtime Environment مثالی از

```java
/* package whatever; // don't place package name */
import java.util.*;
import java.lang.*;
import java.io.*;

// Name of the class has to be "Main" only if the class is public.
public class Ideone
{
    public static void main (String[] args) throws java.lang.Exception
    {
        // your code goes here
    }
}
```
(bluemix.net) Runtime Environment
(bluemix.net) Runtime Environment مثالی از
Virtual Machine vs Container
Component مشخصه های

- Independently Replaceable
- Independently Upgradeable
مشخصه‌های عمومی طراحی مبتنی بر میکروسرویس

• Componentization via services
• Decentralized Governance
• Decentralized Data Management
• Infrastructure Automation
• Evolutionary Design
• …?
مثالی از یک الگوی طراحی برای ESB با میکروسرویس
طراحی مبتنی بر ماشین مجازی

- DescribeInstances
- RunInstances
- StartInstances
- StopInstances
- TerminateInstances
- RebootInstances
- CreateVolume
- AttachVolume
- CreateSnapshot
- AllocateAddress
- ...

Creating a Connection

The first step in accessing EC2 is to create a connection to the service. The recommended way of doing this in boto is:

```python
>>> import boto.ec2
>>> conn = boto.ec2.connect_to_region("us-west-2",
...    aws_access_key_id='<aws access key>',
...    aws_secret_access_key='<aws secret key>')
```

At this point the variable `conn` will point to an EC2Connection object. In this example, the AWS access key and AWS secret key are passed in to the method explicitly. Alternatively, you can set the boto config environment variables and then simply specify which region you want as follows:

```python
>>> conn = boto.ec2.connect_to_region("us-west-2")
```

In either case, `conn` will point to an EC2Connection object which we will use throughout the remainder of this tutorial.

Launching Instances

Possibly, the most important and common task you'll use EC2 for is to launch, stop and terminate instances. In its most primitive form, you can launch an instance as follows:

```python
>>> conn.run_instances('ami-image-id')
```
چند نمونه الگوی طراحی مبتنی بر ماشین مجازی

- Zero Downtime
- Load Balancer
- Resource Replication
- Billing Management System
- Non-Disruptive Service Relocation
- Power Consumption Reduction

Zero Downtime
Power Consumption Reduction
Load Balancer
Resource Replication
Billing Management System
Non-Disruptive Service Relocation
مطالعه بیشتر در

http://crc.aut.ac.ir

http://www.occc.ir

AWS Cloud Design Patterns
cloudpatterns.org
Cloud Computing Patterns
باید تشکر از توجه شما