TSR / NIST – Second Partial (Retake)

This exam consists of 10 multiple choice questions. In every case only one answer is correct. You should answer in a separate sheet. If correctly answered, they contribute 1 point to the exam grade. If incorrectly answered, the contribution is negative: -0.333. So, think carefully your answers.

THEORY

1. In the scope of deployment, this is an example of "component":

X	An independent program that, once running, is one of the active parts in a service. True. That is the meaning of the term "component" that we have used in Unit 4 and Seminar 4.
	A NodeJS module to be used within the programs. False. An application component may be deployed in a container. It is the entity being executed in that container. A NodeJS module is only a small part of a program. The complete program is the component, but not only the module.
	The ZeroMQ middleware. False. ZeroMQ is only a library, not a program/component that may run on its own.
	DNS. False. DNS is a naming service, not a component.

2. The passive replication model...

X	...is unable to handle the Byzantine failure model. True. The Byzantine failure model may only be handled in an appropriate way in the active replication model and using many active replicas to this end.
	...may handle replica failures (in both primary and secondary roles) faster than the active replication model. False. In the active replication model, a replica failure may be easily handled since all replicas behave in the same way. They all play the same role and no service reconfiguration is needed. In the passive replication model a primary failure requires a costly service reconfiguration since another primary replica must be chosen and all clients may need to change their dependences on that primary in some cases.
	...cannot ensure sequential consistency. False. In the passive replication model the primary replica sends the same sequence of updates to every secondary replica. This update order is the same order seen in the primary. As a result of this, every replica may see the same sequence of updates on their state. That kind of consistency is sequential.
	...is unable to handle the stop failure model. False. The stop failure model is the easiest one to be handled. The passive replication model may handle it without many problems. To this end, if there may be up to "f" simultaneous stop failures in the set of replicas, at least "f+1" replicas should exist. One of those remaining replicas will be the new primary after that set of failures.

TSR / NIST – Second Partial (Retake)

3. Considering the constraints of the CAP theorem, when a network partition happens in a sequentially consistent database replicated on all system nodes...

X	…write accesses are not allowed in minor process subgroups. True. This implies a partial loss of availability and consistency in minor subgroups. They remain stopped for write accesses and are not consistent with the state of the greatest subgroup, but in this way such greatest subgroup maintains sequential consistency among its own replicas and it is kept available for all types of client requests.
	…all writes stop, since no updates are allowed while the network is partitioned. False. This implies a complete loss of (write) availability in that database service. Such scenario is not adequate for scalable services.
	…all reads block, since otherwise sequential consistency will be lost. False. This implies a complete loss of (read) availability in that database service. Such scenario is not adequate for scalable services.
	…availability is lost in the subgroup that contains a majority of processes. False. See the explanation in part "b".

4. From these alternatives, which is the best one for improving the scalability of a service?

X	To use the most relaxed consistency model tolerated by the logic of that service. True. As opposed to what has been explained in part "a", a very relaxed consistency model tolerates lazy update propagation. This eliminates the need of synchronisation among replicas. With no synchronisation requirements, the scalability may be maximised.
	To geographically distribute its replicas (for avoiding a single power source) and take voting-based decisions. False. A geographical distribution may require long message propagation times. If decisions require voting from all participating replicas, a lot of synchronisation is demanded in that case. Again, this prevents such service from scaling.
	To use strict consistency among the replicas of that service. False. Strict consistency demands a lot of synchronisation among all replicas. This prevents such service from scaling.
	To deploy its components using Docker. False. Docker automates deployment tasks, but good scalability not only depends on those tasks.

TSR / NIST – Second Partial (Retake)

SEMINARS

5. With the "docker build" command we can...

X	Create a Docker image, taking a Dockerfile as its base. True. That is the goal of the "docker build" command.
	Create a Docker image, taking a Docker container as its base. False. That is done with the "docker commit" command.
	Pause a Docker container, allowing its resumption when needed. False. That is done with the "docker stop" command.
	Generate a Docker container, taking a Docker image as its base. False. That is done with the "docker run" command.

6. In order to implement eventual consistency, we need:

X	Some mechanisms for ensuring state convergence when there are long intervals without any update operation. True. Indeed, that is the definition of eventual consistency.
	A mechanism for ensuring state convergence while the network remains partitioned. False. No state convergence may be achieved while the network remains partitioned if processes may run in that situation.
	A sequencer process. False. A sequencer process is usually needed in non-fast consistency models (e.g., in the sequential model), but eventual consistency is a fast model.
	An immediate propagation of write operations onto every other replica. False. That is the defining characteristic for strict consistency.

7. Which of the following statements about the "docker-compose" command IS FALSE?

X	It is a combination of the "docker build" and "docker run" commands. False. With "docker build" a Docker image is created from a Dockerfile, and with "docker run" a container is started from an image. However, the goal of "docker-compose" is more challenging: to drive the deployment of a complete service consisting of many components in a given host computer.
	By default, its "up" action starts an instance of every component in a given service. True. That is the default goal of the "docker-compose up" command.
	It may be used for applying scale-out or scale-in actions to the components of a given service. True. This can be done with the "docker-compose scale" command.
	It may be used for deploying a composed service in the local host. True. See explanation in part "a".

TSR / NIST – Second Partial (Retake)

8. Which statement is FALSE about "mongod" processes:

X	They are request forwarders. False. That is the role of the "mongos" processes, but not in the "mongod" ones.
	They hold a shard of the MongoDB database. True. In case of using horizontal partitioning, each "mongod" instance manages a different database shard.
	They are placed in server computers. True. They are database servers.
	They may be replaced by a "replica set". True. When replication is considered, each database shard is managed by a different "mongod" replica set.

9. We are writing a NodeJS program with the 'cluster' module. It uses as many workers as twice the amount of processors.

```
var cluster = require('cluster');
var http = require('http');
if (cluster.isMaster) {
  var numReqs = 0;
  var numWorkers = 0;
/* A */
  function messageHandler(msg) {
    numReqs++;
  }
  var numCPUs = require('os').cpus().length;
/* B */
  var numWorkers = numCPUs*2;
  for (var i in cluster.workers)
    cluster.workers[i].on('message', messageHandler);
} else {
  http.Server(function(req, res) {
    res.writeHead(200); res.end('hello world\n');
    process.send({ cmd: 'notify' });
  }).listen(8000);
}
```

Let us assume that those initial workers have been created in part B of the code (not shown) and that the variable "numWorkers" keeps the current amount of workers.
Let us write the following extension (in the place where comment A is shown): If the amount of requests served every 10 ms exceeds the current "numWorkers" value, a new worker should be started. What is the code needed to implement that extension?

X	`setInterval( function() { if (numReqs > numWorkers) { numWorkers++; cluster.fork()}; numReqs=0 }, 10 );` True. This code correctly implements that extension.

	`setInterval( function() { if (numReqs > numWorkers) {`
	`  numWorkers++; process.fork()}; numReqs=0 }, 10 );`
	False. In order to create a worker process, we should use the cluster.fork() operation.

	`setInterval( function() { if (numReqs > numWorkers)`
	`  cluster.fork();}, 10000 );`
	False. There are several errors in this fragment: (1) the numWorkers counter must be increased there, (2) the numReqs counter should be set to zero, and (3) the interval to be used must last 10 ms instead of 10 s.

	`setInterval( function() { if (numReqs > numWorkers)`
	`  process.fork();}, 10 );`
	False. There are several errors in this fragment: (1) the numWorkers counter must be increased there, (2) the numReqs counter should be set to zero, and (3) the intended operation is cluster.fork() instead of process.fork().

10. Let us assume that this Dockerfile is correct (i.e., no error arises when it is processed):

```
FROM exercise10
COPY ./myProgram.js /server.js
EXPOSE 8000 8001 8002
ENTRYPOINT [“node”, “server.js”]
CMD [“type=B”,”id=2”]
```

Which of the following sentences is FALSE?

X	The ENTRYPOINT instruction installs the “node” package on the image that we are building. False. The goal of the ENTRYPOINT instructions is not the installation of a package in the image being built, but the specification of the program to be run when the container is started.
	Other values for “type” and “id” may be passed to the “server.js” program with the “docker run” command without modifying this Dockerfile. True. This is allowed by the CMD instruction.
	There is an "exercise10" Docker image, either in the local or the global repository. True. This is required by the FROM instruction.
	The “server.js” program uses at least three ports. True. This is stated in the EXPOSE-based line.