Τμήμα Ψηφιακών Συστημάτων
Πανεπιστήμιο Πειραιώς

Πληροφοριακά Συστήματα - Extra Εργαστήριο

Χρυσόστομος Συμβουλίδης, simvoul@unipi.gr
Jean-Didier Totow, totow@unipi.gr

Πίνακας περιεχομένων

Scalability

- Η δυνατότητα ενός συστήματος, ενός δικτύου ή μιας υπηρεσίας να διαχειρίζεται αυξανόμενη ποσότητα εργασίας, ή η επέκτασή του δυναμικά προκειμένου να καλύψει πιθανή αύξηση φόρτου εργασίας
- Λέμε ότι ένα σύστημα είναι scalable όταν είναι ικανό να αυξήσει την αποδοτικότητα του αυξάνοντας τους διαθέσιμους πόρους του

Γιατί κάνουμε scaling στις εφαρμογές

- Η πρόβλεψη της χρήσης μιας εφαρμογής ή ενός service αποτελεί μία δύσκολη δοκιμασία
- Ένα σύστημα όσο αυξάνονται οι χρήστες που το χρησιμοποιούν, χρειάζεται περισσότερους πόρους (CPU, Memory, Disk, Bandwidth, …) για να μπορεί να αποδώσει αποτελεσματικά
  - Στη περίπτωση που δεν υπάρχουν διαθέσιμοι πόροι μία υπηρεσία θα παίρνει περισσότερο χρόνο να εξυπηρετήσει τους χρήστες της ή δεν θα μπορεί να τους εξυπηρετήσει καθόλου
- Άρα είναι πολύ χρήσιμο να μπορεί η υπηρεσία να μπορεί να διαχειριστεί τους διαθέσιμους πόρους της με τέτοιο τρόπο ώστε να μπορεί σε κάθε περίπτωση να προσαρμοστεί στις ανάγκες των χρηστών της

Service Level Agreements (SLA)

- Τα Service Level Agreement (SLA) είναι συμβόλαια μεταξύ ενός παρόχου υπηρεσιών και ενός πελάτη τα οποία περιγράφουν συγκεκριμένες απαιτήσεις που πρέπει να πληρούνται (πχ Ποιότητα, Διαθεσιμότητα, κλπ) σχετικά με την υπηρεσία που παρέχεται.
- Χρησιμοποιούνται μετρικές οι οποίες μετράνε αν το SLA παραβιάζεται:
  - Availability: 99,99% uptime
  - Response time: <2 s
  - …

Vertical Scaling

Η διαδικασία κατά την οποία προστίθενται / αφαιρούνται πόροι από μία υπηρεσία ανάλογα με το φόρτο εργασίας που έχει ή για να μπορέσει να κρατήσει σε αποδεκτό επίπεδο την απόδοσή της βάσει του SLA

- **Scale up:** Προστίθενται περισσότεροι πόροι σε ένα ήδη υπάρχον σύστημα.
  Για παράδειγμα σε ένα web service το οποίο χρησιμοποιεί 2xCPU, 4GB RAM και 20GB Storage, μπορεί να προστεθούν ακόμα 2 πυρήνες CPU και 4GB RAM. Τελικά θα λειτουργεί με 4xCPU, 8GB RAM και 20GB Storage.
- **Scale down:** Αν οι πόροι που έχουν δοθεί σε ένα σύστημα δεν αξιοποιούνται αφαιρούνται κάποιοι για να αποφευχθούν επιπλέον χρεώσεις. Σε αναλογία με το προηγούμενο παράδειγμα, μπορεί πλέον το service να λειτουργεί με 2xCPU, 2GB RAM και 20GB Storage.

Horizontal Scaling

Η διαδικασία κατά την οποία προστίθενται / αφαιρούνται πόροι από μία υπηρεσία ανάλογα με το φόρτο εργασίας που έχει ή για να μπορέσει να κρατήσει σε αποδεκτό επίπεδο την απόδοσή της βάσει του SLA

- **Scale out:** Προστίθενται περισσότεροι υπολογιστικοί κόμβοι (computing nodes) στην υπηρεσία (πχ ένα ακόμα VM, ένα ακόμα container κλπ) προκειμένου να μοιραστεί ο φόρτος εργασίας σε περισσότερους
- **Scale in:** Αφαιρούνται αχρείαστοι υπολογιστικοί κόμβοι

Manual scaling

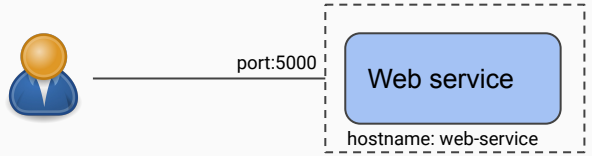
- **Manual scaling:** Η διαδικασία κατά την οποία επιπρόσθετοι πόροι προστίθενται μετά από παρέμβαση του χρήστη.
- Γίνεται μέσω ενός εργαλείου διαχείρισης container (container orchestrator).
- Τα εργαλεία διαχείρισης αυτά παρέχουν στους χρήστες τους εργαλεία (εντολές) για την προσθήκη / αφαίρεση μνήμης (memory), επεξεργαστικής ισχύος (cpu), ή/και πανομοιότυπων container (replicas) κλπ.
- Επίσης διαχειρίζονται και το φόρτο εργασίας (load balancing)

Containers orchestrator:

- Docker swarm
- Kubernetes
- κλπ.

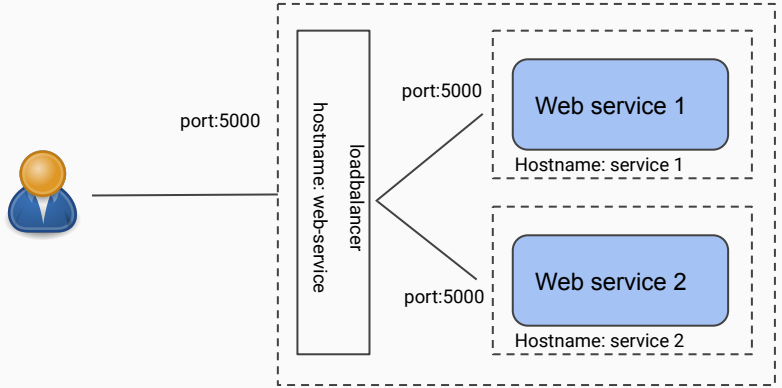
Load balancing

- Εξισορρόπηση φορτίου (Load balancing): Η διαδικασία κατά την οποία μοιράζεται ο φόρτος εργασίας ενός συστήματος σε πανομοιότυπες υπηρεσίες (replicas)
- Μία ρέπλικα αποτελεί ουσιαστικά ένα αντίγραφο του / της ήδη υπάρχοντος / υπάρχουσας συστήματος / υπηρεσίας
- Load balancing εφαρμόζεται σε περιπτώσεις horizontal (scale out)



Δύο βασικοί αλγόριθμοι load balancing:

- Static load balancing: Που δεν λαμβάνεται υπόψη η κατάσταση (state) της υπηρεσίας: (Ex.: round robin)
- Dynamic load balancing: Εδώ η κατάσταση στην οποία βρίσκεται η υπηρεσία, λαμβάνεται υπόψη



Χαρακτηριστικά:

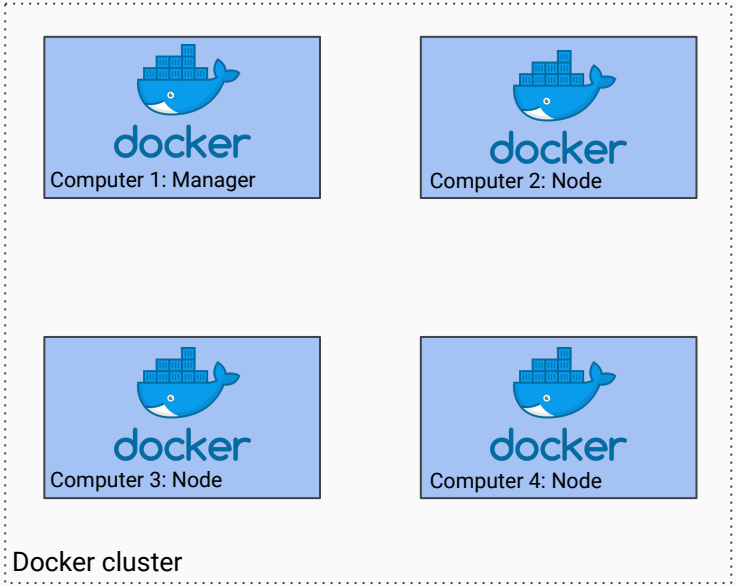
- Ασύμμετρος φόρτος Asymmetric load
- Προστασία από επιθέσεις κατανεμημένης άρνησης εξυπηρέτησης (DDoS attack)
- Http caching
- Firewall

Docker swarm

Το Docker μπορεί να τρέξει και σε swarm mode ώστε να διαχειριστεί ένα cluster απο docker engine.

Χαρακτηριστικά του Swarm mode:

- Από το docker 1.12, το docker swarm mode είναι integrated με το docker
- Cluster management integrated with Docker Engine
- Decentralized design
- Declarative service model
- Scaling
- Service discovery
- Load balancing
- κλπ.



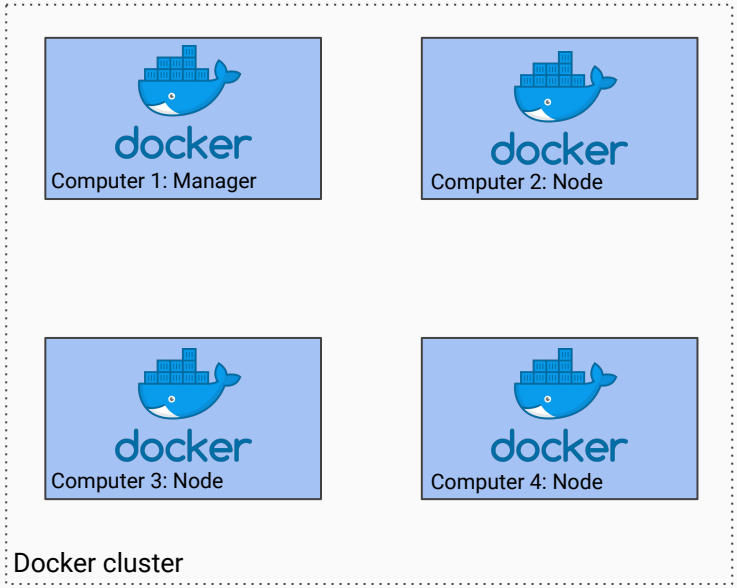
Docker swarm activation & running services

- **`docker swarm init --advertise-addr <hostname/IP>`** : αρχικοποίηση ενός docker swarm
- **`docker swarm join --token <TOKEN> <hostname:port>`** : ανάθεση ενός cluster στο docker assigning docker to a docker cluster
- **`docker swarm leave`**: έξοδος από το swarm

Αφού ενεργοποιηθεί το docker swarm ολα τα docker engine του cluster ενοποιούνται και τα container γίνονται deploy σε όλα τα διαθέσιμα μηχανήματα.

Οτιδήποτε έχει γίνει deploy στο swarm, λέγεται service

- **`docker service create`**: δημιουργία υπηρεσίας
- **`docker service ls`**: εμφάνιση λίστας διαθέσιμων υπηρεσιών
- **`docker service logs`**: εμφάνιση log μιας υπηρεσίας



Running docker-compose file on swarm mode

- Μόνο docker-compose με version >= 3 υποστηρίζεται

Για να γίνει deploy ένα service:

`docker stack deploy -c docker-compose.yaml <name>`: γίνονται deploy τα service από ένα αρχείο docker-compose

`docker stack rm <name>` : αφαίρεση του deployed service

`docker service scale service_name=N` : γίνεται replicate το deployed service

```
ID                  NAME                MODE                REPLICAS            IMAGE                                          PORTS
g66v8cv8c4u8        lab6_exporter       replicated          1/1                 jdtotow/exporter:latest                        *:55684->55684/tcp
01a6714ixma3        lab6_flask          replicated          1/1                 jdtotow/flask:latest                           *:5000->5000/tcp
htjlfljomrx5        lab6_grafana        replicated          1/1                 grafana/grafana:latest                         *:3000->3000/tcp
qi9p92nauuzq        lab6_logstash       replicated          1/1                 docker.elastic.co/logstash/logstash:6.4.3      *:8081->8081/tcp
1r997ad0938v        lab6_mongodb        replicated          1/1                 mongo:latest                                   *:27017->27017/tcp
x4k772nag95j        lab6_prometheus     replicated          1/1                 cuigh/prometheus:latest                        *:9090->9090/tcp
gfvm9jlo4fmb        lab6_rabbitmq       replicated          1/1                 jdtotow/rabbitmq:latest                        *:5671-5672->5671-5672/tcp,
pv7cvj8qbgky        lab6_slalite        replicated          0/1                 fermenreq/slalite:test                         *:8090->8090/tcp
```

Auto scaling

- Με το auto scaling δίνεται αυτόματα το έναυσμα για να γίνει replicate ένα service όταν ένα συμβαίνει κάποιο γεγονός στο σύστημα.
- Ένα τέτοιο συμβάν τις περισσότερες των περιπτώσεων, αφορά μια παράβαση ενός ενεργού SLA.

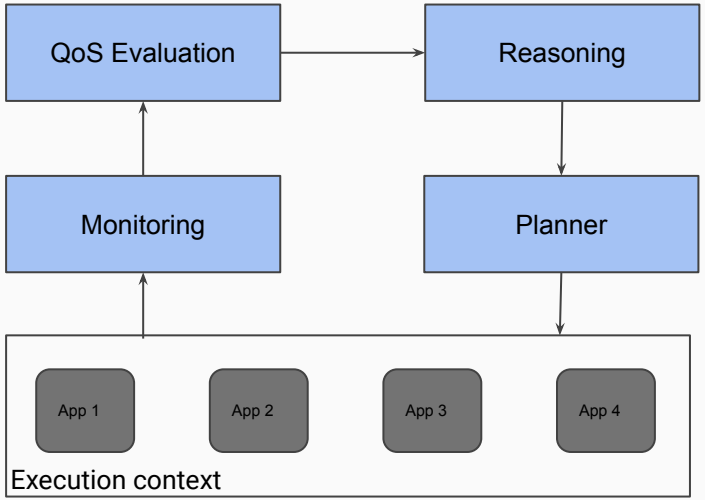
Παραδείγματα:

- Χρόνος απόκρισης > τιμή ,
- throughput < τιμή
- αριθμός χρηστών > τιμή
- κλπ.

Ανάγκες εφαρμογής:

- Ανάγκη για distribution
- Δυνατότητα για παρακολούθηση (Monitoring)
- Δημιουργία προφίλ της εφαρμογής

Auto scaling model



- Γενικό πλαίσιο εκτέλεσης: docker engine, kubernetes, cloudiator
- Monitoring: collects & expose metrics
- QoS Evaluation: Αξιολόγηση της συμμόρφωσης με το SLA βάσει των πραγματικών τιμών
- Reasoning: Επιλογή των αλλαγών που πρέπει να γίνουν
- Planner: Σχεδιασμός του deployment, ανανέωση ή αφαίρεσή του

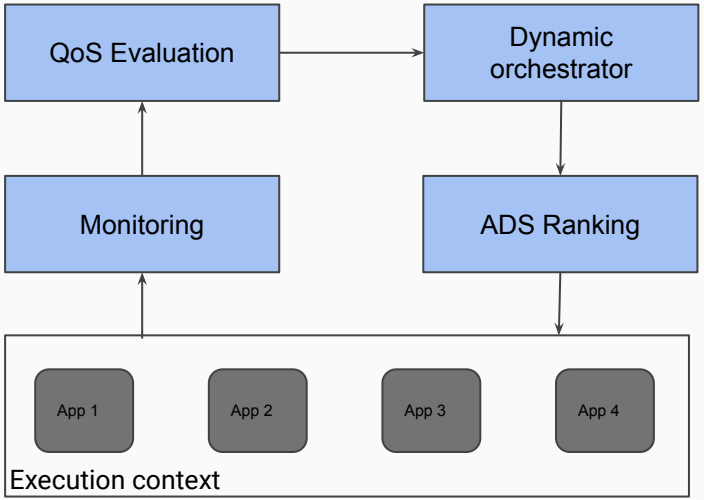
BigDataStack

BigDataStack
Holistic stack for big data applications and operations

Data-driven infrastructure for data intensive application.

Χαρακτηριστικά:

- Auto-scalable
- Βασισμένο στο kubernetes



Demo

Demo components

- Flask service
- MongoDB
- Prometheus
- Grafana
- Swarm visualizer