Das Testbed wird in folgender Arbeit genau beschrieben: Kontroll- und Messumgebung für Netzwerkprototypen.

Eine kurze Anleitung befindet sich in svn/testbed/README.

Komponenten

Das Testbed besteht aus zwei Programmkomponenten: Einem *Master* und mindestens einem *Client*. Der *Master* koordiniert das Experiment und ist die Schnittstelle für den Benutzer um das Experiment steuern zu können. Auf jedem Host, der für das Experiment benutzt wird, läuft ein *Client*. Dieser ist für die Steuerung der Prototypen zuständig, die auf diesem Host laufen. Die *Clients* werden von dem *Master* gesteuert.

Konfiguration

SQL:

Zusätzlich, zu der aus der Simulation bekannten Konfiguration, kommen noch folgende Tabellen hinzu:

- *prototypes*: Hier werden die Prototypen spezifiziert. Zu einem *command* (dieser *command* wird über die Tabelle *node_groups* referenziert) können bis zu zwei Prototypen angegeben werden (jeweils für 32- und 64-Bit). Mit *libs_path* können libs angegeben werden, die bei der Ausführung dem Prototypen per *LD_LIBRARY_PATH* zur Verfügung stehen. Mit *data_path* werden Dateien angegeben, die in das Arbeitsverzeichnis des Prototypen kopiert werden.
- *hosts*: Hier werden die zur Verfügung stehenden Hosts eingetragen. Die Hosts müssen per SSH erreichbar sein (Zugangsdaten: *user_name*, *key_file*). *max_prototypes* gibt an, wie viele Prototypen maximal auf diesem Host ausgeführt werden dürfen. Die Portrange für die Prototypen wird mit *usable_ports_start* und *usable_ports_end* angegeben. *working_dir* gibt das Arbeitsverzeichnis auf diesem Host an.
- *node_host_placement*: Diese Konfiguration ist optional. Hier können Relationen zwischen nodegroups angegeben werden, welche Auswirkungen auf die Zuordnung der Knoten zu den Hosts haben. Zur Verfügung stehen *bound* und *exclusion*. Wird zB nodegroup**A** mit *exclusion* zu nodegroup**B** eingetragen, werden Knoten der beiden Gruppen nicht auf demselben Host platziert.
- *node_data*: Diese Konfiguration ist optional. Hier können zu einer nodegroup zusätzliche Dateien angegeben werden, unabhängig von dem zugeordnetem Prototypen.

PlanetLab/GermanLab

Für PlanetLab/GermanLab steht ein Script zur Verfügung, das die Eintragung der nodes in die Tabelle *hosts* automatisch übernimmt. Ausführen mit: `./PLCAPI.py`. Als *host working dir* sollte ein Verzeichnis gewählt werden, welches eindeutig zu dem Benutzer zugeordnet werden kann.

Programmeinstellungen:

Das Verhalten des Testbeds kann über einige Parameter angepasst werden. In der Datei `testbed/base/Constants.py` befinden sich viele Parameter. Unter anderem kann das explizite Überprüfen der HOST-Keys für SSH Verbindungen deaktiviert werden (`SSH_HOST_KEY_CHECKING=False`), die Anzahl der WorkerThreads angepasst werden (`WORKER_THREADS`) oder die (mindest)Vorlaufzeit für den Startzeitpunkt eines Experiments festgelegt werden (`START_EXPERIMENT_LEADTIME_SECONDS`).

Experiment: Ablauf

1. Konfiguration des Experiments (siehe Abschnitt *Konfiguration*)
2. `make` ausführen, dann Master starten (siehe Abschnitt *Programmstart*)
3. Discovery
    - Der *Master* überprüft die Hosts, welche Verfügbar sind und stellt deren Eigenschaften fest. Hier werden auch die aktuell verfügbaren Ressourcen ermittelt (Arbeitsspeicher, Festplatte, …).
4. Deployment
    - Der Benutzer wird nun gefragt, ob er möglichst viele Hosts verwenden möchte, oder möglich wenig. Dies hat Einfluss auf die automatische Zuordnung der nodes zu den Hosts. Wählt der Benutzer die Option `less`, werden so wenig Hosts wie möglich verwendet. Dh pro Host werden viele nodes zugeordnet. Wählt der Benutzer `many`, werden möglichst alle verfügbaren Hosts verwendet. Letztere Option sorgt wahrscheinlich für bessere Ergebnisse, ist jedoch gleichzeitig auch aufwendiger, da viel mehr Hosts vorbereitet werden müssen.
    - Nach der Zuordnung der nodes werden die Hosts vorbereitet. Der *Master* kopiert die *Clients* sowie die Prototypen auf die Hosts.
    - Danach werden die *Clients* gestartet sodass diese weitere Vorbereitungen treffen können.
    - Der Master bis alle Clients bereit sind und fragt dann den Benutzer nach dem Startzeitpunkt des Experiments.

    - Sobald der Master den Clients den Startzeitpunkt mittgeteilt hat, kann der Master beendet werden.
5. Monitoring
    - Während des Experiments muss der Master nicht aktiv sein.
    - Der Zustand der Clients kann mit dem Master überprüft werden.
6. Postprocessing
    - Sobald das Experiment vorbei ist, kontaktiert der *Master* die Hosts um festzustellen, welche mit der Nachbereitung der Datenbanken der nodes bereits fertig ist. Findet der *Master* solche, lädt er die Datenbank vom Host und fügt diese in die eigene Datenbank ein.
    - Nachdem die Datenbanken aller Hosts kopiert wurden, befinden sich die Mess/Log/Dump-Daten aller Knoten in der zentralen Datenbank.
    - Neben der Zusammenfassung der Datenbanken können noch weitere Daten von den Hosts geladen werden. Hierzu mehr im Abschnitt *Auswertung*.

Programmstart:

Der Master wird über das `run.sh` Skript gestartet. Es müssen mindestens diese Parameter gesetzt sein: `./run.sh -d database.db -e experiment` (analog zum Simulator).

Das Logging kann mit `-v`, `-vv`, … konfiguriert werden. Bei höheren Logleveln ist die Verwendung des Logservers zu empfehlen. Mehr hierzu in Abschnitt *Logging*.

(Die Parameter *--debug* und *--profile* sind nur für interne Tests relevant)

Logging:

Die Logausgaben des Masters können an einen Logserver gesendet werden. Der Maser wird dazu mit `--logserver` gestartet.

Der Logserver wird mit `python -m testbed.scripts.LoggingServer -p 7777` gestartet. Dieser zeigt alle Logs an, die lokal an Port TCP 7777 gesendet werden. Der Master ist entsprechend konfiguriert und nutzt den gleichen Port.

Auswertung:

Es stehen mehrere Möglichkeiten zur Verfügung nach einem Experiment weitere Daten von den Hosts zu laden:

- nach nodeID(s)
- nach nodeGroup(s)
- nach hostID(s)

Die Daten werden im Verzeichnis collectedData gespeichert.

Verzeichnisstruktur:

Master:

Wird das Arbeitsverzeichnis initialisiert (python -m testbed.init), werden folgende Dateien/Order erstellt: - *Makefile* zum erstellen der Datenbanken - *run.sh* zum Starten des Masters - *SQL* Order der die fürs Testet angepassten SQL Daten beinhaltet

Während der Laufzeit des Masters, werden noch weitere Dateien/Ordner erzeugt: - *CollectedData* enthält die von den Hosts geladenen Dateien - *experiment.db* die exp Datenbank - *experiment_node.db* die Datenbank für die Knoten - *knownHosts* enthält die ssh fingerprints der Hosts - *python-static* der statisch gebundene Python Interpreter für die Hosts - *sshConnections* wird für die Verwaltung der SSH-Master Connections benötigt

Host:

Im Arbeitsverzeichnis des Hosts wird der Python-Interpreter sowie das inspect.py Script. Diese sind unabhängig vom Experiment. Für jedes Experiment wird nun ein eigener Unterordner angelegt. Darin befinden sich die Arbeitsverzeichnisse der nodes sowie weitere Daten.