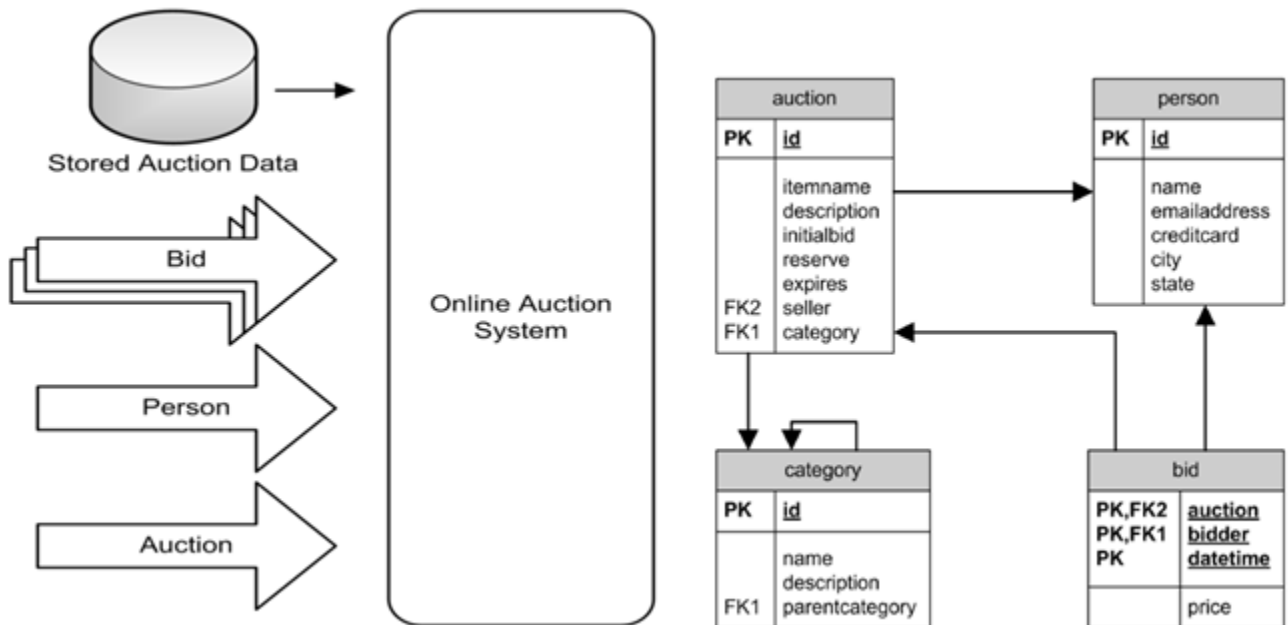


Getting Started with Nexmark

Nexmark simulates an online auction setting and was originally designed as an benchmark for the Niagara streaming system (see <http://datalab.cs.pdx.edu/niagara/NEXMark/>).



In this scenario there are three streams Person, Auction and Bid and stored information (Category). At every time, new users can register themselves in the system and create new auctions. Each user can also generate bids for auction. The Schema can be found in the picture above.

As default: Each time a client connects new (to one of the four streams) a new simulation is started (with reproducible data). To get consistent data, connections to the other three streams, will use the same simulation. So e.g., if you connect to a person stream and have no other connections to the server, the simulation start from the beginning. If you connect to another stream, e.g., bid, and you are still connected to the person stream, the bid stream fits to the person stream. If you connect to the bid stream after you disconnected from the person stream, a new simulation will be started. This behavior can be changed with an start up option.

We adapted this scenario for Odysseus and allow to configure the scenario.

Get and Start Nexmark Generator

We recommend using the docker version. Here all dependencies are already provided inside the docker-image

You could download nexmark from dockerhub (for windows and macos you could use the [Docker Desktop](#)):

```
docker pull odysseusol/nexmark
docker run -d -p 65440-65443:65440-65443 odysseusol/nexmark

#if you want to configure nexmark (Remark: Change YOURLOCALFOLDER to you folder e.g. "c:/volume/nexmark", that
contains NEXMarkGeneratorConfiguration.properties, absolute paths are necessary here)
docker run -d -v "YOURLOCALFOLDER:/var/lib/nexmark" -p 65440-65443:65440-65443 odysseusol/nexmark

#if you want to run nexmark even after restarts and configure nexmark (Remark: Change YOURLOCALFOLDER with you
folder e.g. "c:/volume/nexmark")
docker run -d --restart unless-stopped -v "YOURLOCALFOLDER:/var/lib/nexmark" -p 65440-65443:65440-65443
odysseusol/nexmark
```

Remark: If you want to configure in docker environments you could use the example files from: <https://odysseus.informatik.uni-oldenburg.de/download/nexmark/config/>. Only the file named NEXMarkGeneratorConfiguration.properties will be used. So change the name or adapt the file if you want to use the SLOW version.

You will see that the benchmark opens 4 servers on ports 65440 - 65443 where you can connect to.

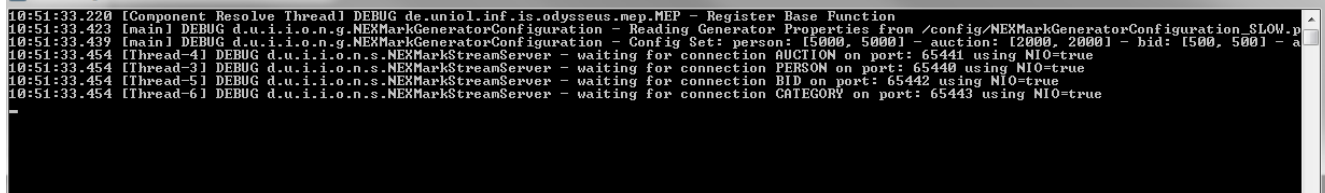
Now Odysseus can use the nexmark sources. Each time an Odysseus query connects to one of the servers a new scenario is started, i.e. the same users and the same auctions are generated, so the same queries will always create the same results. The created events are consistent, e.g. there will be no bid for an auction that has not been created.

An alternative is to use the download version:

Go to <http://odysseus.informatik.uni-oldenburg.de/download/nexmark/> and download a version that fits to your operating system (e.g. nexmark.win32.win32.x86_64.zip for a 64bit Windows).

Nexmark needs at least Java 11.

Unzip the archive and start the "nexmark.exe" within the nexmark folder. This should look like in the following:

A screenshot of a terminal window with a black background and white text. The logs show the following: 10:51:33.220 [Component Resolve Thread] DEBUG de.uniold.inf.is.odysseus.mgp.MEP - Register Base Function; 10:51:33.423 [main] DEBUG d.u.i.i.o.n.g.NEXMarkGeneratorConfiguration - Reading Generator Properties from /config/NEXMarkGeneratorConfiguration_SLOW.p; 10:51:33.439 [main] DEBUG d.u.i.i.o.n.g.NEXMarkGeneratorConfiguration - Config Set: person: [5000, 5000] - auction: [2000, 2000] - bid: [500, 500] - a; 10:51:33.454 [Thread-4] DEBUG d.u.i.i.o.n.s.NEXMarkStreamServer - waiting for connection AUCTION on port: 65441 using NIO=true; 10:51:33.454 [Thread-3] DEBUG d.u.i.i.o.n.s.NEXMarkStreamServer - waiting for connection PERSON on port: 65440 using NIO=true; 10:51:33.454 [Thread-5] DEBUG d.u.i.i.o.n.s.NEXMarkStreamServer - waiting for connection BID on port: 65442 using NIO=true; 10:51:33.454 [Thread-6] DEBUG d.u.i.i.o.n.s.NEXMarkStreamServer - waiting for connection CATEGORY on port: 65443 using NIO=true.

Error Windows (10 or 11):

Remark: Especially, if there are multiple JDK installed, you could get an error like "No exit data available" and Nexmark does not start. This is typically because the wrong JDK is bound. In this case you could open the nexmark.ini File and add two lines before **-vmargs (Change path to your jvm.dll location for Java 11)**

```
-vm
C:/Program Files/Java/jdk-11.0.8/bin/server/jvm.dll
```

Configure Nexmark

Nexmark has some standard configuration that will be used in our examples. The configuration can be changed in `nexmark.ini`.

The following parameters can be adapted:

- `pr`: This is the starting port of the first server (Auction). The next servers (PERSON, BID, CATEGORY) will get the next 3 ports.
- `gcf`: This parameter references a file that describes the behaviour of the nexmark server, e.g. how long is the delay between different elements in the stream. We deliver two standard configurations that can be found in the plugin `de.uniold.inf.is.odysseus.nexmark`:
 - `/config/NEXMarkGeneratorConfiguration_SLOW.properties`: Here elements are delivered with a rather slow rate. This should be used when creating queries.
 - `/config/NEXMarkGeneratorConfiguration.properties`: Here elements are created with a higher rate. This can be used to test, if the application scales.
- `useGlobalGenerator`: If this flag is set Nexmark will start the simulation with the first access from any client and resumes the simulation until the program is stopped explicitly.

Nexmark Configuration File

When using docker, you will need to bind a volume with -v (see example above)

In the following code block is an example of a nexmark configuration (`/config/NEXMarkGeneratorConfiguration_SLOW.properties`). All time elements are in milliseconds.

The following elements can be used:

- `minDistBetweenPersons`: How much time must pass, before a new person event can be created.
- `maxDistBetweenPersons`: What is the maximum time that is allowed to pass, before a new person is created.
 - Hints:
 - `max` must be higher or equal than `min`
 - if `max == min` this is the rate at which the elements are created, else there will be a random value between `min` and `max`.
 - the values are application time, e.g. the events will get a time stamp corresponding to the `min/max` value. This could be different from system time values!
- `min/maxDistBetweenAuctions/Bids`: The same as above, but for auctions and bids.
- `acceleratorFactor`: Typically, application time and system time are the same. So if there is 5 seconds defined as waiting time, there will be created elements each 5 seconds in real time (if the server is capable to, but this should be no problem with 5 seconds 😊). To get the same application time stamp but speed up processing, this parameter can be used.
- `min/maxAuctionDuration`: Each auction has a time at which bids can be posed. This values give the min and the max value, that will be used for each auction.
- `min/maxTimeBetweenBursts`: To allow times with higher data rates (more elements are send), bursts can be used. This value give the min and max waiting time between bursts. This value will not use `acceleratorFactor`, because it realtime and not applicationtime specific
- `min/maxBurstDuration`: How long will bursts takes.
- `burstAcclerationFactor`: How many more elements will be send, during a burst phase.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>

<!-- Konfiguration des NEXMarkGenerators -->
<!-- Zeitangaben in MS -->

<entry key="minDistBetweenPersons">5000</entry>
<entry key="maxDistBetweenPersons">5000</entry>

<entry key="minDistBetweenAuctions">2000</entry>
<entry key="maxDistBetweenAuctions">2000</entry>

<entry key="minDistBetweenBids">500</entry>
<entry key="maxDistBetweenBids">500</entry>

<!-- Faktor um den die Zeit beschleunigt werden soll. -->
<entry key="accelerationFactor">1</entry>

<!-- wie lange Auktionen offen bleiben sollen -->
<entry key="minAuctionDuration">600000</entry>
<entry key="maxAuctionDuration">600000</entry>

<!-- Zeit zwischen zwei Bursts (MIN und MAX) in ms -->
<!-- Ein Wert von 0 bei beiden simuliert ohne Bursts. -->
<!-- Diese Zeit wird nicht vom "accelerationFactor" beeinflusst -->
<entry key="minTimeBetweenBursts">0</entry>
<entry key="maxTimeBetweenBursts">0</entry>

<!-- Dauer eines Bursts -->
<!-- Ein Wert von 0 bei beiden simuliert ohne Bursts. -->
<!-- Diese Zeit wird nicht vom "accelerationFactor" beeinflusst -->
<entry key="minBurstDuration">0</entry>
<entry key="maxBurstDuration">0</entry>

<!-- Beschleunigungsfaktor waehrend eines Bursts -->
<entry key="burstAccelerationFactor">0</entry>

</properties>
```