Optimizing Radeon VRAM usage

Lauri Kasanen

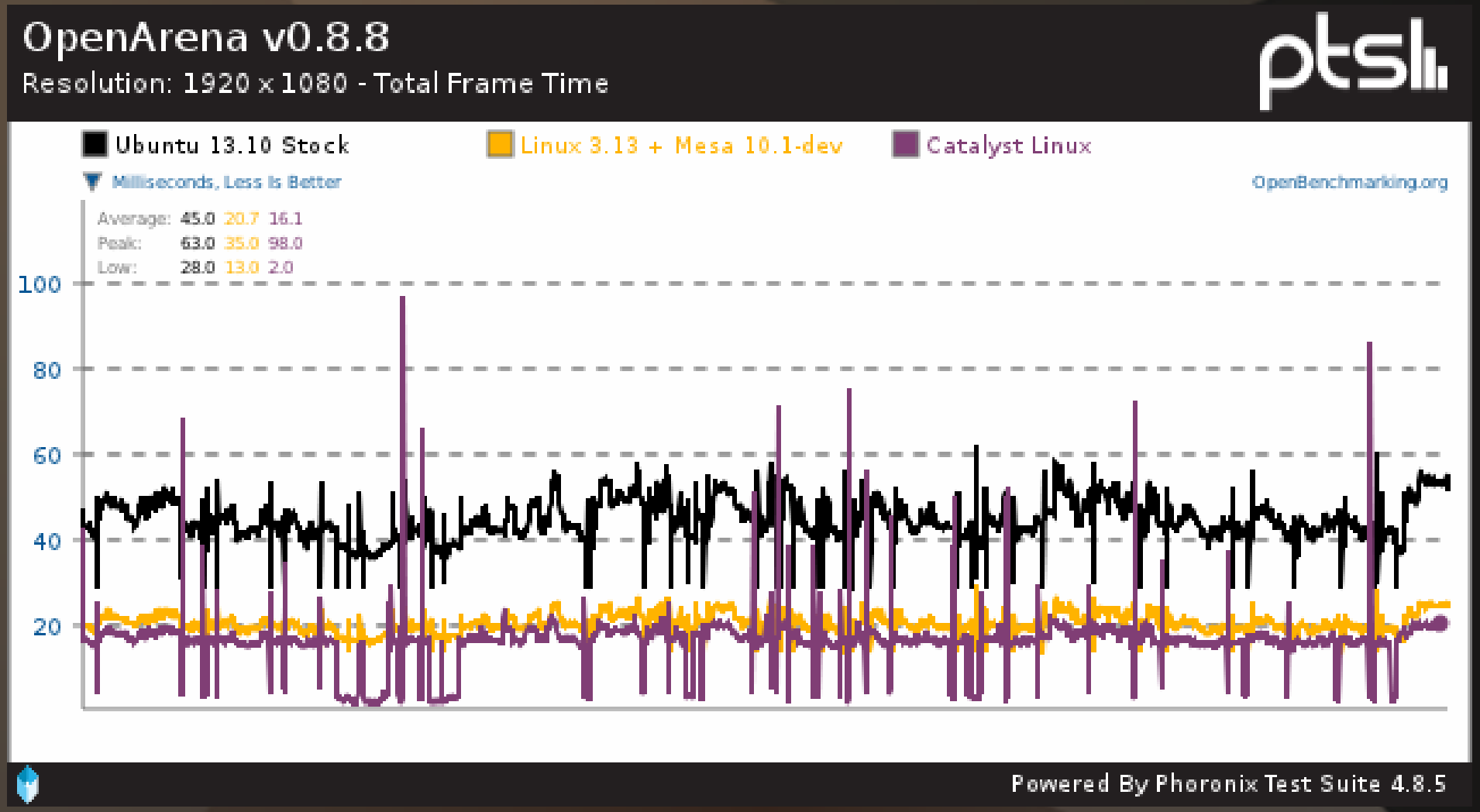
Agenda

- Lähtökohta
- Frame-time-mittaus
- Fragmentaatio
- Tutkimussuunta
- Neuroverkot, koulutus
- Ohjelmisto
- Tulokset
- Kysymyksiä?

Lähtökohta

- Puskureiden tarpeetonta edestakaista liikennettä (ping-pong) ei ollut juuri tutkittu
- Muutaman bugiraportin mukaan nykyinen koodi aiheuttaa surkeaa suorituskykyä rajatapauksissa
- Puskureiden sijainnin optimoinnilla on mahdollisuus parantaa sekä suorituskykyä että FPS:n tasaisuutta (smoothness), muistipaineen alaisuudessa

Frame-time-mittaus



Fragmentaatio

- Tehokkaaseen muistinkäyttöön liittyy olennaisesti fragmentaatio (pirstaloituminen)
- Mikäli fragmentaatiota olisi mahdollista helposti vähentää, saaden näin hukkatilaa käyttöön, muistipaineeseen jouduttaisiin harvemmin
- Oletus: allokoimalla suuret puskurit muistin toisesta päästä, ja pienet toisesta, syntyvä fragmentaatio aiheuttaa vähemmän uudelleenkäyttöön sopimattomia välejä

Tutkimussuunta

- Oletus on, että puskuriin liittyvissä tilastoissa on yhteys puskurin tärkeyteen
- Pitämällä tärkeät puskurit VRAM-muistissa pitäisi ping-pongin vähentyä
- Tällainen yhteys on todennäköisesti epälineaarinen ja monimutkainen
- Sellaisten yhteyksien löytämiseen käytetään tavallisesti tekoälyä

Kerätyt tilastot

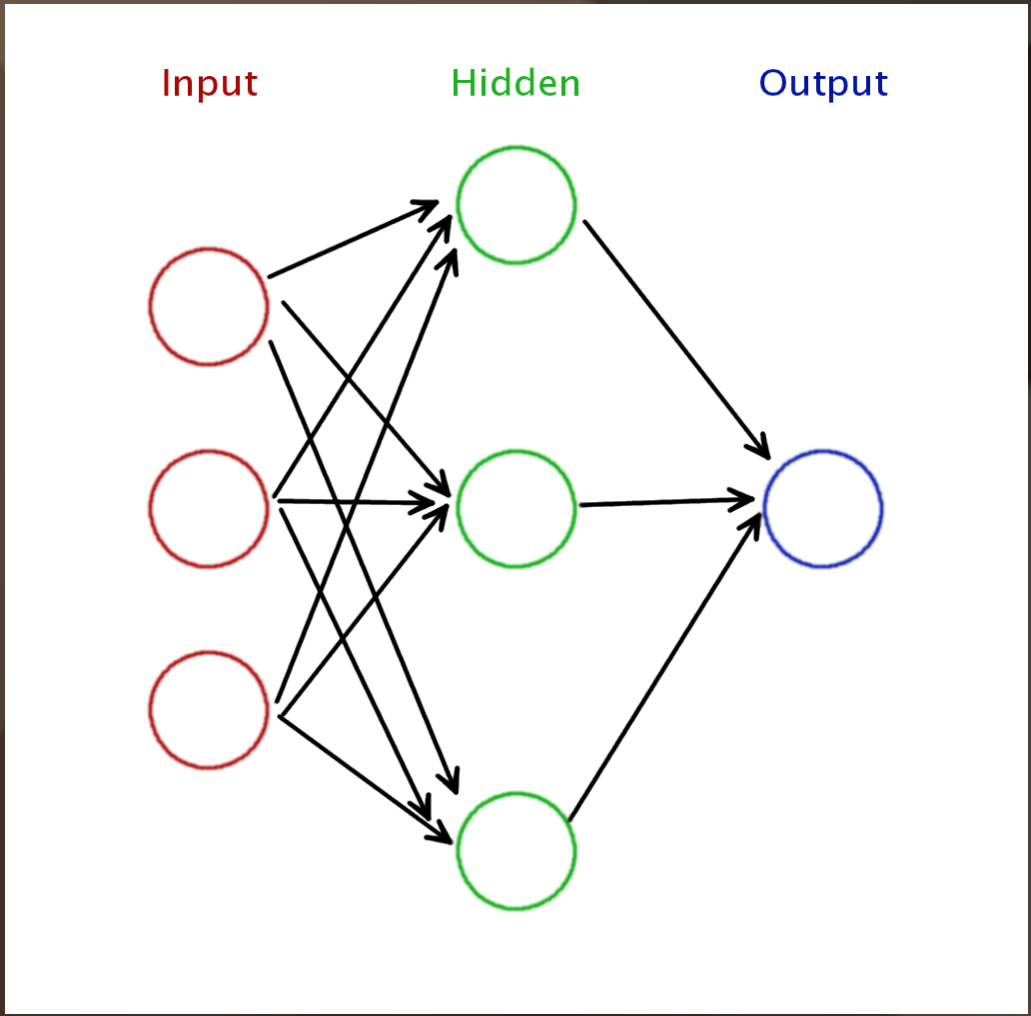
- Kirjoituksen, luvun, ja CPU-operaatioden määrä ja aika viimeisestä sellaisesta
- Puskurin koko
- VRAM-koko
- Onko puskuri tärkeä (MSAA, syvyys)

Neuroverkot

- Yleisin tekoälyn muoto
- Useita erilaisia verkkorakenteita
- Useita erilaisia koulutustapoja

- Rakenteeksi valittiin yleisin, multi-level perceptron

Multi-level perceptron



Koulutus

- Yllättäen yksikään yleisesti käytetyistä koulutusmetodeista ei soveltunut tähän ongelmaan (supervised, unsupervised, competitive, reinforcement)
- Päätettiin käyttää geneettistä metodia, mikä ei kouluta (yhtä) verkkoa, vaan kehittää evoluution avulla ratkaisua tuhansien ratkaisujen populaatiosta
- Hienosäätö Monte-Carlo-metodilla

Ohjelmisto

- Erilaisten lähestymistapojen tehokkuuden mittaamista varten kehitettiin muistisimulaattori, sekä muuta ohjelmistoa
- Muistijälkiä kerättiin suuresta määrästä pelejä ja sovelluksia
- Ajamalla tallennettu jälki simulaattorin läpi voitiin mitata useita eri arvoja (mm. fragmentaation määrä, puskurisiirtojen määrä)

Muistijälki

```
cpu op buffer 7 at 3 ms
create buffer 8 at 3 ms (8 bytes)
cpu op buffer 8 at 3 ms
create buffer 9 at 3 ms (48 bytes)
cpu op buffer 9 at 3 ms
create buffer 10 at 3 ms (48 bytes)
cpu op buffer 10 at 3 ms
create buffer 11 at 3 ms (24 bytes)
cpu op buffer 11 at 3 ms
create buffer 12 at 3 ms (32 bytes)
cpu op buffer 12 at 3 ms
create buffer 13 at 3 ms (8 bytes)
cpu op buffer 13 at 3 ms
create buffer 14 at 3 ms (16384 bytes)
create buffer 15 at 3 ms (65536 bytes)
create buffer 16 at 3 ms (16384 bytes)
write buffer 16 at 3 ms
create buffer 17 at 3 ms (4096 bytes, high priority)
destroy buffer 17 at 3 ms
destroy buffer 17 at 3 ms
destroy buffer 14 at 3 ms
cpu op buffer 7 at 11 ms
```

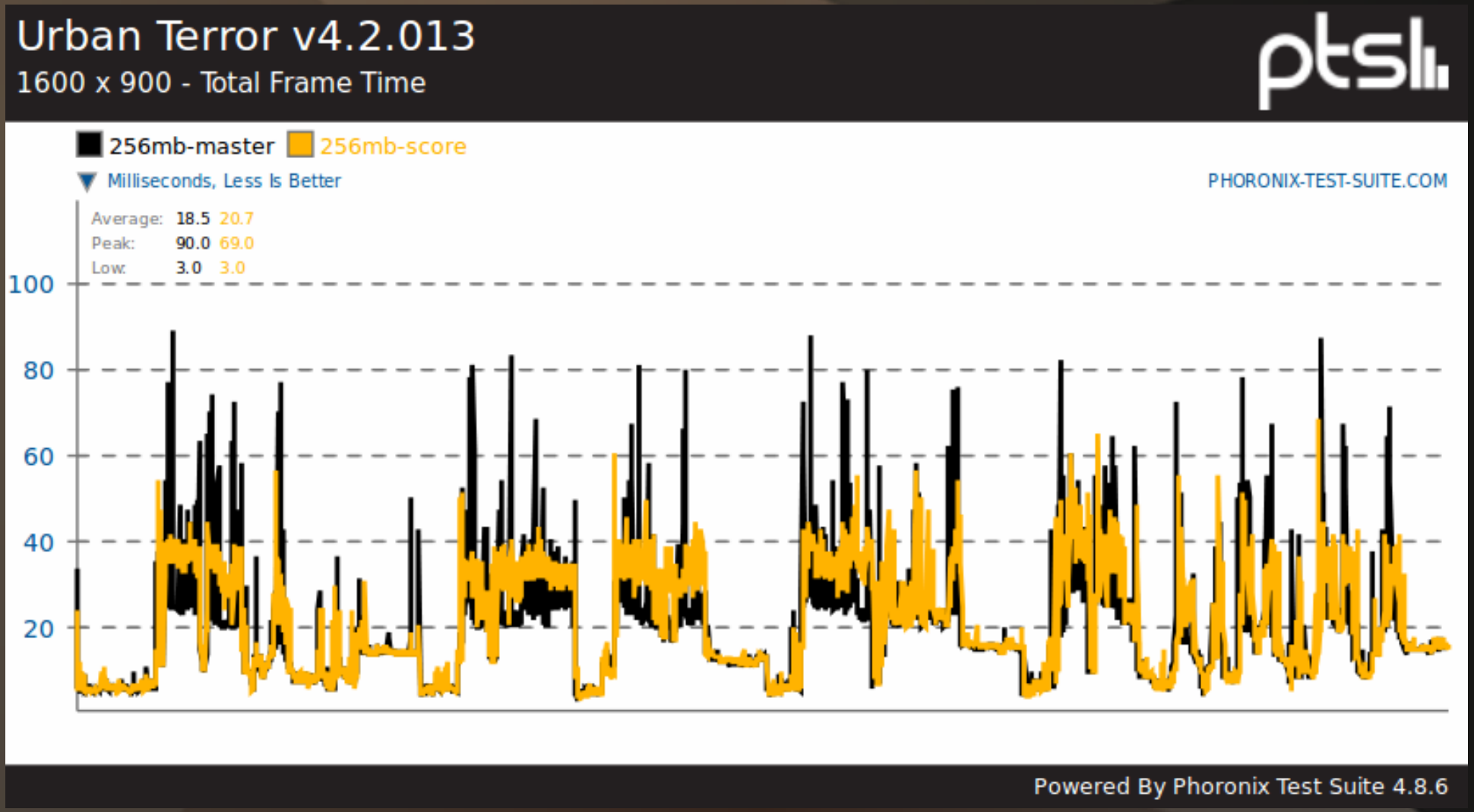
Tulokset: fragmentaatio

- Testattu kaksisuuntainen allokaattori onnistui vähentämään ping-pongia jopa 20%
- Allokaattori on saatavilla uusimmassa Linux-ytimessä (3.15)

Tulokset: tekoäly

- Useimmissa testatuissa sovelluksissa suorituskyky parani 1-2% muistipaineen alla
- Testeissä, jotka tukivat frame-time-mittausta, huiput olivat matalampia ja niitä oli selvästi vähemmän (smoothness)

Frame-time-mittaus



Kysymyksiä?

No squirrel.

Squirrel.