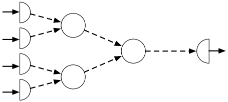
FLICK: Application specific network functions for datacentres



Richard G. Clegg Imperial College
Work joint with the Network-as-a-Service project:
Imperial College: Abdul Alim, Paolo Costa, Luo Mai, Peter Pietzuch, Lukas Rupprecht, Alexander Wolf
Cambridge: Jon Crowcroft, Anil Madhavapeddy, Andrew Moore, Richard Mortier, Nik Sultana
Nottingham: Derek McAuley, Masoud Koleini, Carlos Oviedo

Talk to Coseners/MSN 2015

Application-specific network functions

Problem

Modern datacentres have many application-specific network functions: load-balancers, cacheing, aggregation...
Written from scratch in low-level programming languages.
No function isolation or sharing of resources.

Application-specific network functions

Problem

Modern datacentres have many application-specific network functions: load-balancers, cacheing, aggregation...
Written from scratch in low-level programming languages.
No function isolation or sharing of resources.

Core concept: Same box, many apps processing TCP streams at layer 7. Work on streams of data items from apps (e.g. key-value pairs, memcached responses) message not packet oriented.

Application-specific network functions

Problem

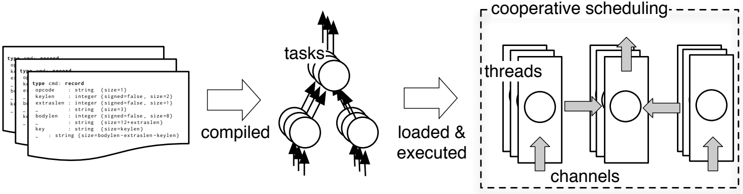
Modern datacentres have many application-specific network functions: load-balancers, cacheing, aggregation...
Written from scratch in low-level programming languages.
No function isolation or sharing of resources.

Core concept: Same box, many apps processing TCP streams at layer 7. Work on streams of data items from apps (e.g. key-value pairs, memcached responses) message not packet oriented.

Solution – FLICK

A domain specific language allowing fast specification of network functions.
A platform for running compiled FLICK programs giving performance and isolation on shared resources.

FLICK overview



```
type cmd: record
 opcode    : string  {size=1}
 keylen    : integer {signed=false, size=2}
 extraslen : integer {signed=false, size=1}
 _         : string  {size=3}
 bodylen   : integer {signed=false, size=8}
 _         : string  {size=12+extraslen}
 key       : string  {size=keylen}
 _   : string {size=bodylen-extraslen-keylen}
```

compiled

tasks

loaded & executed

cooperative scheduling

threads

channels

FLICK programs

1

FLICK task graphs

2

FLICK platform

3

- Programs – Domain specific HLL. “Safe by design”.
- Task graphs – takes care of task/data parallelism.
- Platform – scheduling and memory management.

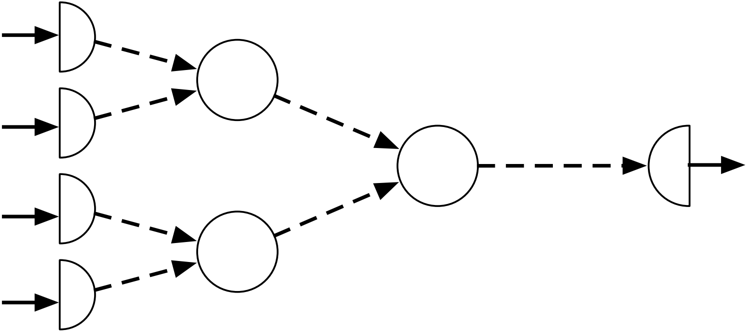
FLICK's tricks – the language

```
type cmd: record
 opcode    : string  {size=1}
 keylen    : integer {signed=false, size=2}
 extraslen : integer {signed=false, size=1}
 _         : string  {size=3}
 bodylen   : integer {signed=false, size=8}
 _         : string  {size=12+extraslen}
 key       : string  {size=keylen}
 _         : string  {size=bodylen-extraslen-keylen}

proc Memcached:
     (cmd/cmd client, [cmd/cmd] backends)
 global cache := empty_dict
 backends => update_cache(cache) => client
 client => test_cache(client, backends, cache)
```

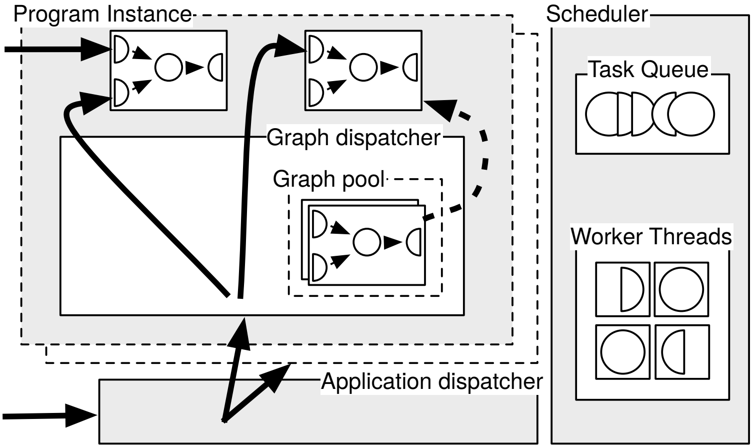
- "Safe-by-design" – small data items, light processing.
- Non Turing complete language.
- Type system implies serialisation/deserialisation.
- Processes application semantics.

FLICK's tricks – the task graph



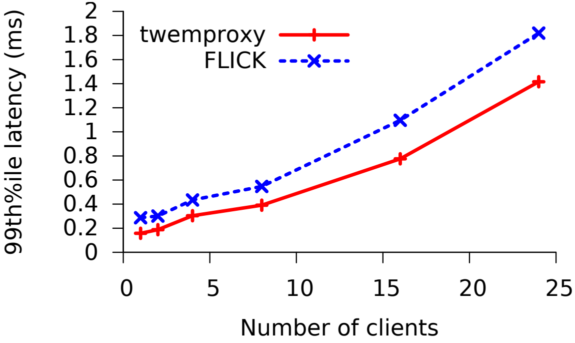
- Task Graph – App specific DAG of independently schedulable tasks.
- Tasks – process streams in batches of one or more "data units". Yield after a small time limit ($\sim$100μs).
- Tasks take advantage of task and data parallelism.
- I/O tasks convert wire format to/from app specific data items. Processing tasks "do the work".

FLICK's tricks – the platform



- Virtual machine for implementing task graphs.
- Handle scheduling and worker threads.
- Instantiate task graphs to process new streams.

Performance – memcached example



- Penalty of generalisation is extremely low.
- Initial results promising – ongoing work.

Conclusions/Future Work

Conclusions

- FLICK language – developers express network applications in a high level.
- FLICK platform – performant, safe implementation on real hardware.

Future work

- Integrate with DPDK/mtcp (userspace) for better performance.
- Hardware offloading to NetFPGA.
- SDN for control of data to/from FLICK platform.