Kamailio in the ITSP: Changing Winds

...

Alex Balashov
Evariste Systems LLC
Athens, Georgia, USA
http://www.evaristesys.com/

Evariste Systems?

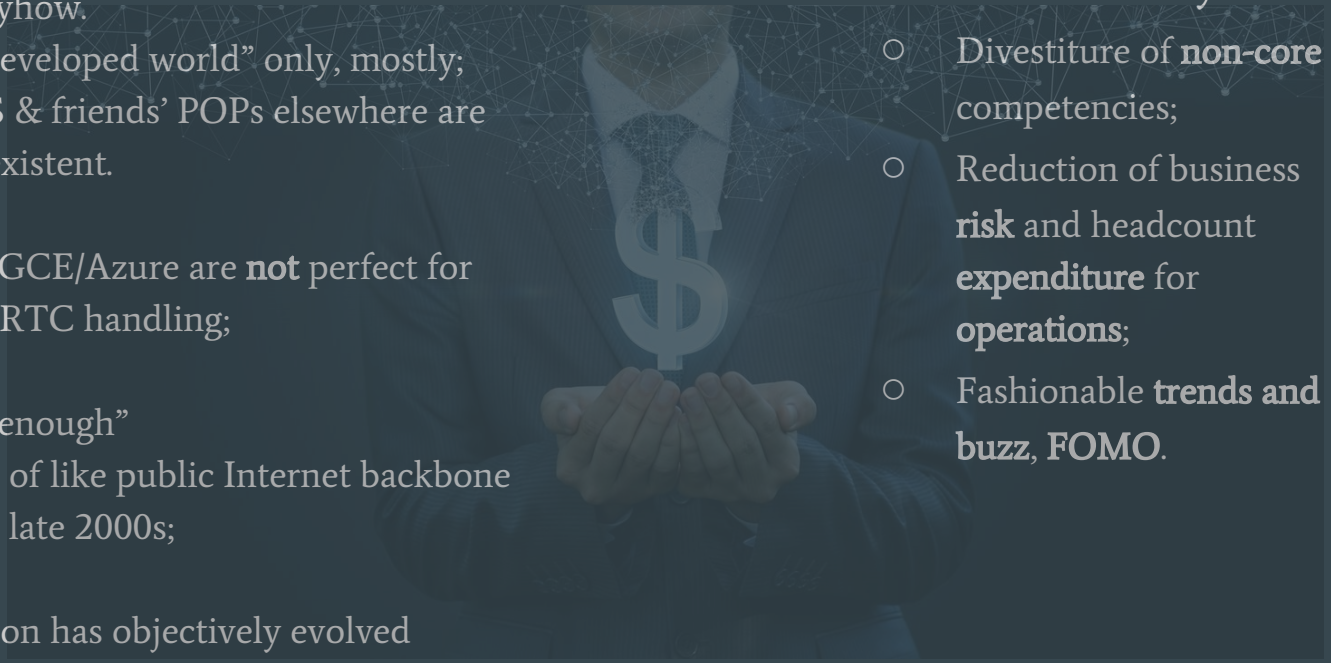
- Based in **Athens, Georgia**, USA (university town close to Atlanta);
- **Kamailio consultancy** from 2007;
- Vendor of **CSRP** (Class 4 routing product) based on Kamailio:
  - http://www.csrpswitch.com/
- Blog on **Kamailio technical topics**:
  - http://www.evaristesys.com/blog/
- Provider of SIP and Kamailio **training**;
- Kamailio project contributor, advocate.

Conventional wisdom about ITSP infrastructure (2000s, early 2010s):

- Big, serious and real-time;
- Dedicated and ample hardware for best economies of scale;
- Cloud and virtualisation are hostile to “media performance” and “real-time” anything;
  - Poor call quality and the rest.
- Real-time communications are special and require artisanal, nuanced infrastructure approach.

Nevertheless:

- **Big shift:** ITSPs embracing major cloud services anyhow.
  - In “developed world” only, mostly; AWS & friends’ POPs elsewhere are nonexistent.
- AWS EC2/GCE/Azure are **not** perfect for media and RTC handling;
- “Just good enough”
  - Kind of like public Internet backbone from late 2000s;
- Virtualisation has objectively evolved
  - Almost first-class CPU tenant;
  - Close “to the metal”.



- CxO suite sold by:
  - Divestiture of **non-core** competencies;
  - Reduction of business **risk** and headcount **expenditure** for **operations**;
  - Fashionable **trends and buzz**, **FOMO**.

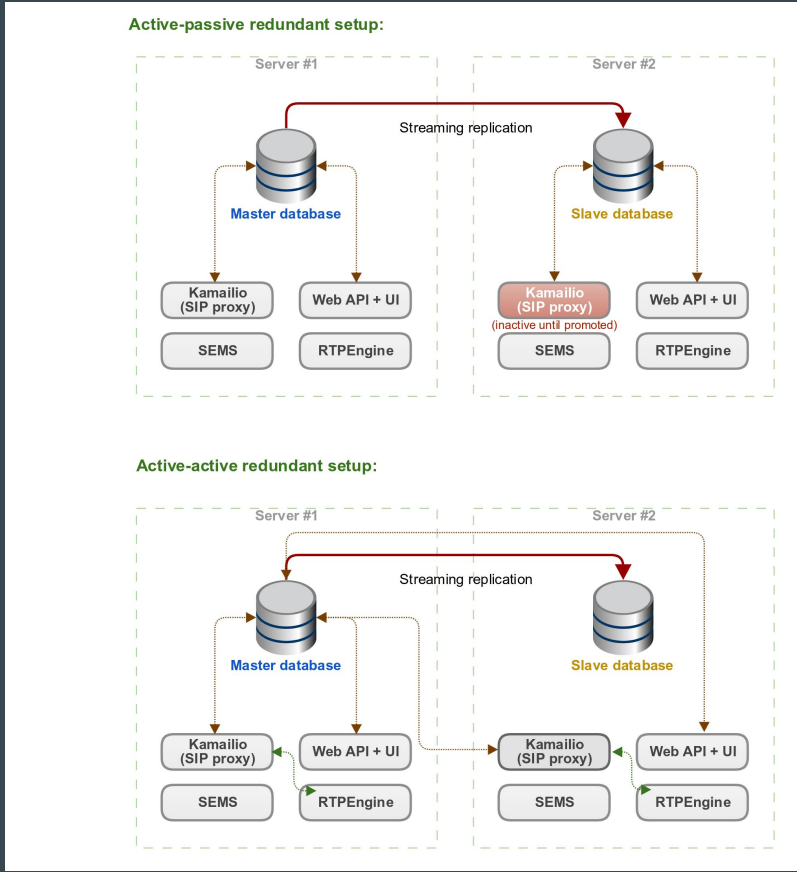
What the businesspeople want:

Fire the **system admins** and reduce **operations** headcount.

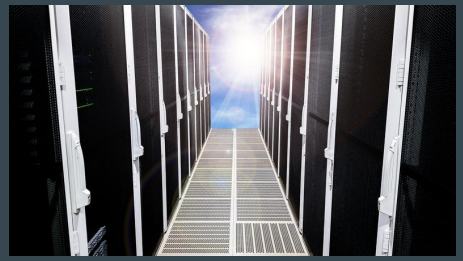
The “big fat box” model of telecoms infrastructure:

- Big, well-resourced physical server (leased dedicated or owned);
  - Typical stats: 4+ cores, 32+ GB of RAM, SSDs, Gigabit LAN.
- Own administration, responsibility for redundancy and upkeep
  - Requires **people** (system admins, data centre remote hands, etc.) for care and feeding;
  - Engineering, storage, etc. mostly local concern, which can be bad without big CAPEX;
- Services provider infrastructure and applications **highly centralised**.
  - Big database;
  - Lots of IOps;
  - Lots of RAM;
  - Central proxy;
  - Aggregation;
  - Focus on optimising throughput.

Typical “big box” data centre architecture (CSRP):



“Big fat box” to cloud:



- **Cannot** simply migrate “big fat box” architecture as-is!

- Hosting on someone else’s hypervisor/infrastructure **is not cloud**; it’s just hosted;
  - To develop cloud-native services, you must “**get**” cloud;
  - **Cloud platform pricing** will not support this approach;
    - Will increase OPEX;
    - Not savings;
    - Big instances are **very expensive.**

- Cloud-native service & application characteristics:
  - Distributed;
  - Dynamic discovery / self-assembly;
  - Elasticity;
  - Dimensions fitted to instances and componentry of cloud.

“Big fat box” to cloud (cont’d):

- Typically requires extensive **re-architecture** of highly centralised applications;
- Just breaking big, centralised components (e.g. into containers) for distribution is not enough;
- You still need [lots of] people to build and run this! Skill sets:
  - Linux sysadmin folk traditions;
  - Updated for modern “cloud” DevOps;
    - Orchestration (Ansible, Salt, Puppet, Chef, etc.);
    - Discovery and synchronisation (e.g. Consul, Serf, Kubernetes, etcd, Redis, Route53, etc, etc.);
    - Cloud platform APIs and automation;
    - Idiosyncrasies of cloud platform (networking, limitations, economics of instances).
  - True to the name: more “dev” to go with “ops”.

Unexpected factors for ITSPs:

- Big instances are **very** expensive — cloud providers really, *really* want you to buy lots of smaller ones;
- Often invisible and non-obvious resource constraints:
  - PPS and bandwidth limits;
  - Not necessarily published;
  - **Backbone** and **transit** transfer limits.
- Occasional scheduling/contention/hypervisor issues;
- Network and reachability issues as artifices of cloud product rather than technological limitations;
- **All of this** has a-la-carte solutions and becomes a line item on your bill!
- Will you save money?
  - Maybe; maybe the opposite.
- Will you reduce risk and improve availability?
  - Probably not, but shape of problem is different.

Kamailio features complementary to cloud-native dev.:

- Hot reloading:
  - dispatcher
  - rtpengine (sets defined in DB)
  - RPC

- Fallback to mostly stateless relay (except for hop-by-hop messages);

- DMQ
  - **dmq_usrloc** and dialog replication
  - Database-synced approaches aren't so good for cloud due to bandwidth/backbone contsraints;

- Flexible logging;

- Flexible invocation and environment for containerised execution;

- **http_async_client** for HTTP REST interactions;

- Options to support 1-to-1 NAT and advertising of public addresses:
  - `listen=udp:x.x.x.x:5060 advertise y.y.y.y:5060`
  - `rtpengine -i external/10.x.x.x!56.1.2.3`

Thanks!

Please find me if you have further questions, or visit: http://www.evaristesys.com/