Standards and the Next Generation of Cloud

(or: How I Stopped Worrying and Learned to Love Software Standards)

craig mcluckie (@cmcluck)
product guy at Google
Sarah Novotny (@sarahnovotny)
community wonk at Google

A thought: Andy Grove and disruption

- "Only the paranoid survive"
- a book about understanding disruption
- 4 current disruptive forces
  - containers
  - cloud
  - IoT
  - XaaS

one way to think the evolution of cloud

- mode 1: ‘commodity cloud’
    - *useful abstractions on commodity infrastructure*
- mode 2: ‘traditional enterprise’ cloud.
    - mode 2.1 -- ‘departmental apps’.
    - mode 2.2 -- ‘integrated applications’.
    - *outsource infrastructure operations*
- mode 3: the ‘hyperscale cloud’.
    - *patterns architected for internet scale infrastructure*

why enterprise cares about mode 3

- starting point: every business needs to reimagine itself as a software company
  - … okay, so become a software company?
- next: open source is eating the software world
  - … okay, so become an open source software company?
- next: scale is exploding
  - … okay, so become an ‘internet scale’ open source software company?

… so you probably care about mode 3

- only one practical way to achieve internet scale computing: **'cloud native'**
- co-evolved in many places: Google, Facebook, Twitter, …
- So #GIFEE (or #FIFEE, #TIFEE, …) helps

- container packaged
  - predictable deployment; efficient resource isolation
- dynamically scheduled
  - radically higher QoS and efficiency; radically lower ops cost
- micro-services oriented
  - radically higher reuse; easier to extend

in getting to 2 billion containers a week...

Google had to build a few things

- Linux containers support: cgroups
  - support efficient resource isolation for process groups
  - contributed to Linux by Google (2006)
- schedulers: Borg and it’s successor Omega
- micro-services naming/discovery
- efficient RPC framework
- build and CI/CD systems
- developer tooling and debugging
- etc, etc, etc

which make us an internet company

the community could go 2 ways to deliver those things

‘vertical integration’

… or … ‘decoupled stacks’

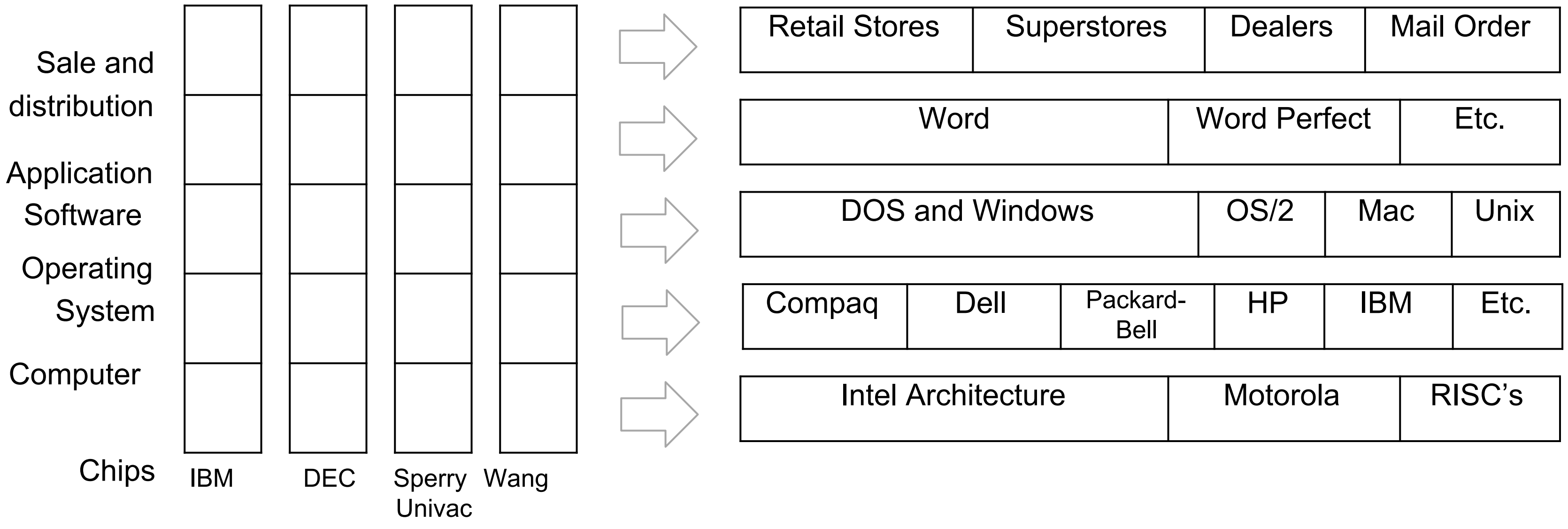
A thought: Andy Grove and specialization

"I tend to believe Mark Twain hit it on the head when he said, "Put all of your eggs in one basket and WATCH THAT BASKET."

– **Andrew S. Grove, Only the Paranoid Survive**

back to Andy Grove

Reproduced from 'Only the Paranoid Survive'

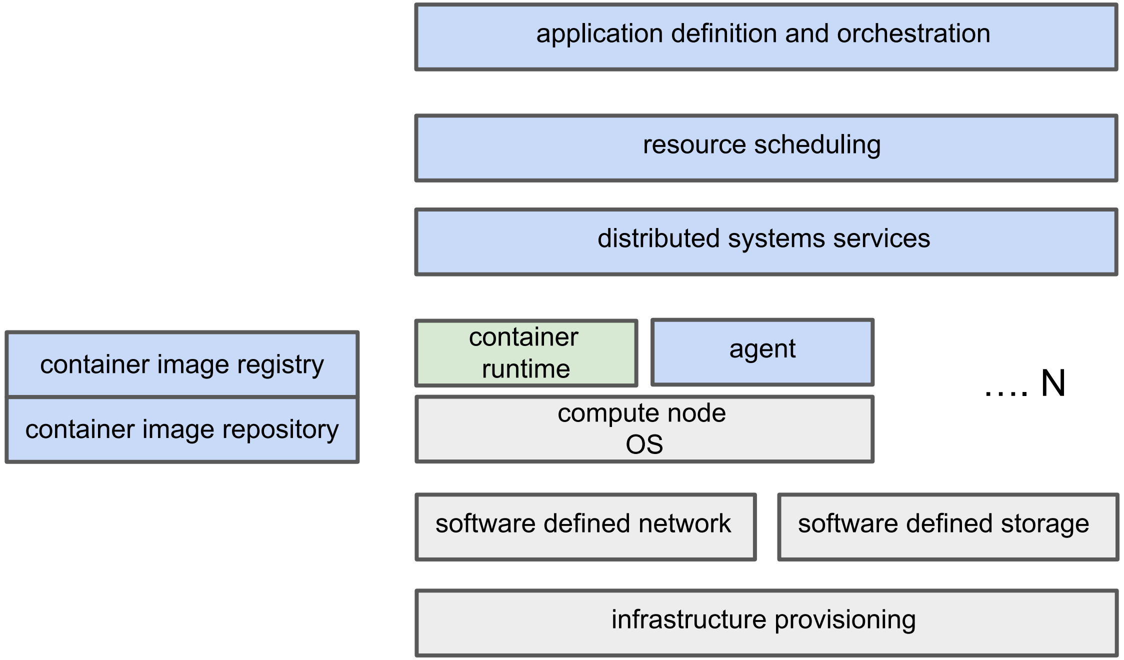


verticalization will fail:
standardization of interfaces is crucial

- an ecosystem needs specialization
- more options are better than fewer
- **a vendor shouldn’t have to implement a whole stack**

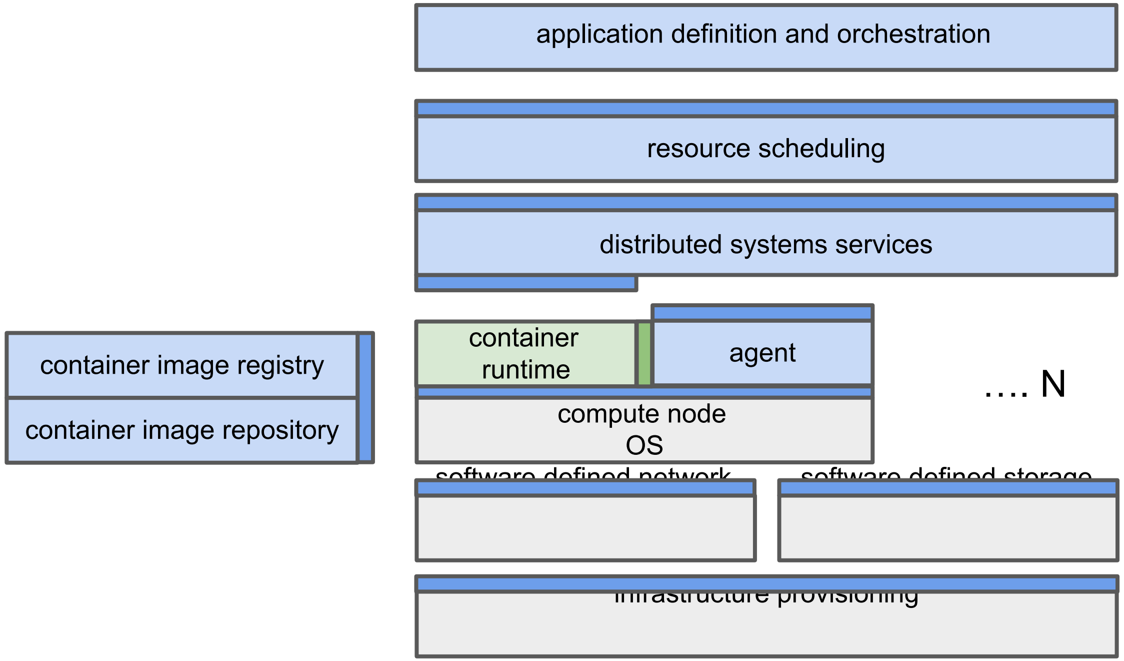
solution

so we need a foundation to deliver whole stack(s)



solution

so we need a foundation to deliver whole stack(s)



a foundation makes sense

CLOUD NATIVE
COMPUTING FOUNDATION

- assemble harmonized sets of technologies
- qualify reference architectures for interoperability
- identify and fill gaps
- create a safe place for the industry to engage and contribute

but we want it to be a great foundation

- address some potential shortcomings
  - be coherent -- hold technical opinion
  - be accountable -- give end users ‘teeth’
  - be consistent -- avoid semantic drift

- address through governance structure
  - balanced authority -- ‘separation of church and state’

how to achieve coherency: technical leadership

- technical oversight committee
- independent, elected group
  - modeled on the supreme court
  - chosen for lifetime contributions
- representation from end-user committee
- hold technical opinion, decide what is ‘cloud native’

how to achieve accountability: end user committee

- independent of vendors
- representative of users
- technical committee is accountable to end user committee

how to achieve consistency: code driven standards

- avoid the tyranny of academic standards: lead with code
- establish semantic standard through reference implementation
- promote API standard over time once you know what works
- demonstrate semantic consistency through conformance testing
  - good: API conformance
  - better: behavioral conformance
  - best: common code

a call to action

CNCF is a different sort of foundation…

- broad vendor support: BC (business committee)
- strong sense of technical identity: TOC (technical oversight committee)
- **strong commitment to end user: EUC (end user committee)**

we are actively recruiting end users for the foundation please visit http:

//cncf.io for details