# Open API 전략 및 동향

김 마이클 mike.kim@sk.com
SK planet / Platform architecture 팀장

# 김 마이클

현) SK planet
플랫폼 아키텍쳐 팀장 (2013~)

- Symantec. Corp (Milpitas)
- Deloitte Consulting (S.F.)
- Agilent Technologies (Santa Clara)
- Compuware Corp. (Colorado Springs)

- Architecture, DB design,
  Web, Windows

- mike.kim@sk.com
  mkim@consultant.com

# Open API

… sets of technologies that enable websites to interact with each other by using REST, SOAP, JavaScript and other web technologies. While its possibilities aren't limited to web-based applications, it's becoming an increasing trend in so-called Web 2.0 applications.

- Wikipedia -

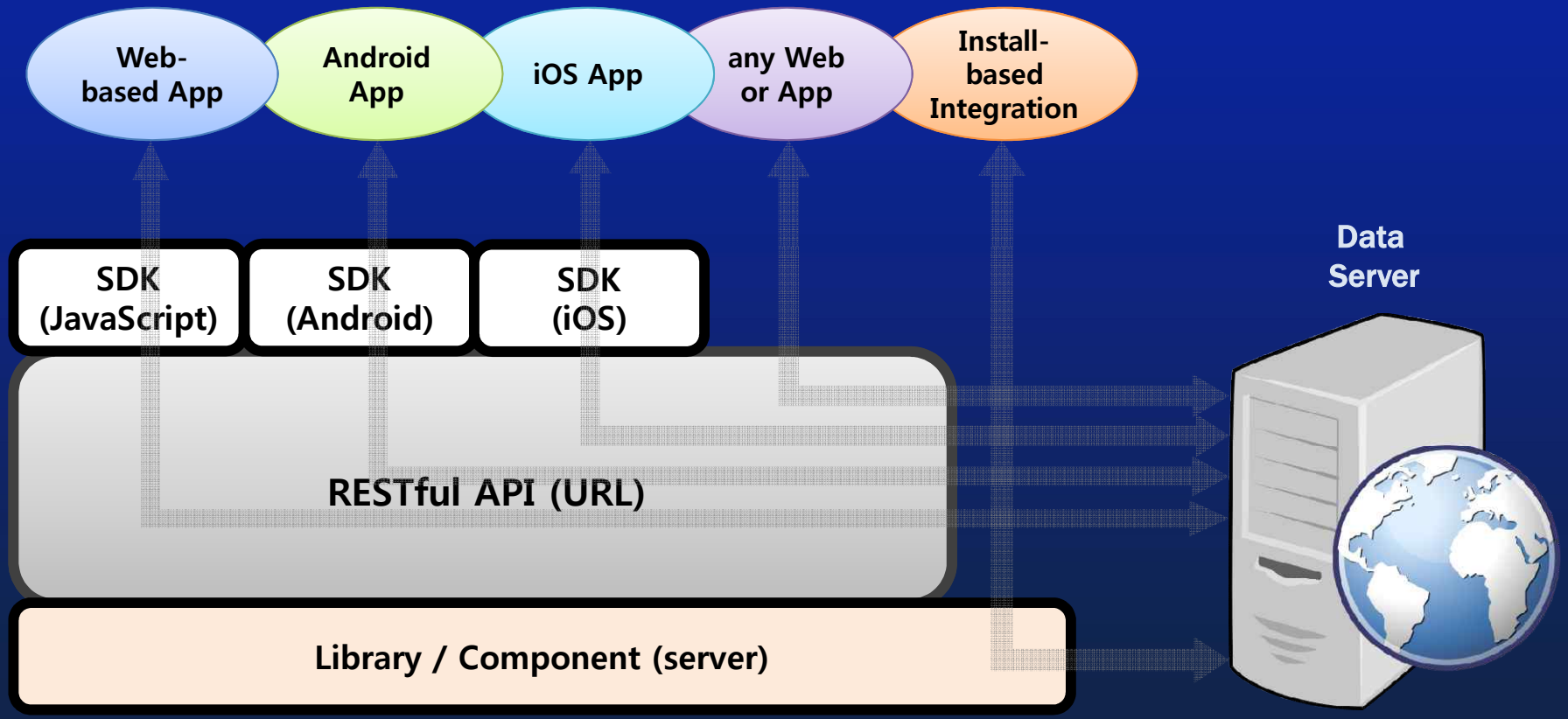
# Open API

… sets of technologies that enable websites to interact with each other by using REST, SOAP, JavaScript and other web technologies. While its possibilities aren't limited to web-based applications, it's becoming an increasing trend in so-called Web 2.0 applications.

- Wikipedia -

- **Current & permanent definition?**
  - **Still evolving**
- **Open**
  - **Open to public only ?**
    **Same set of API could open to partner / internal**
  - **Open standard protocol / format**

# 통상적인 API Packaging



# SOA vs. API management

Isn't it just a new name for SOA?

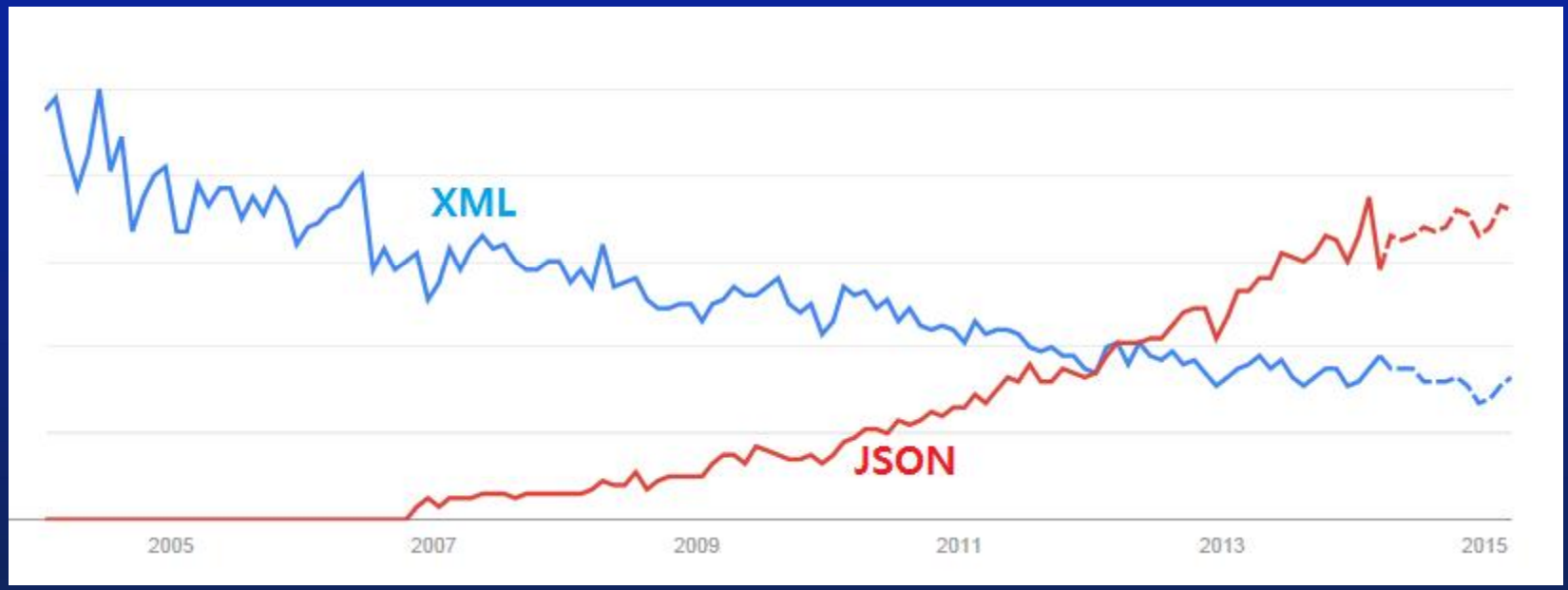
Not entirely. SOA strategies mostly target internal users; open Web APIs target mostly external partners. So API management requires developer portals, key management, and metering and billing facilities that SOA management never provided.

Forrester Wave: API management platforms, Q1 2013

# 최근 API 의 De facto standard

- JSON - XML 보다 적은 데이터 양
- REST – URL 을 통한 간단한 호출 방식
- OAuth 2.0 – 사용자 Account 정보를 3rd party 애플리케이션에 넘기지 않고도 Authentication / Authorization

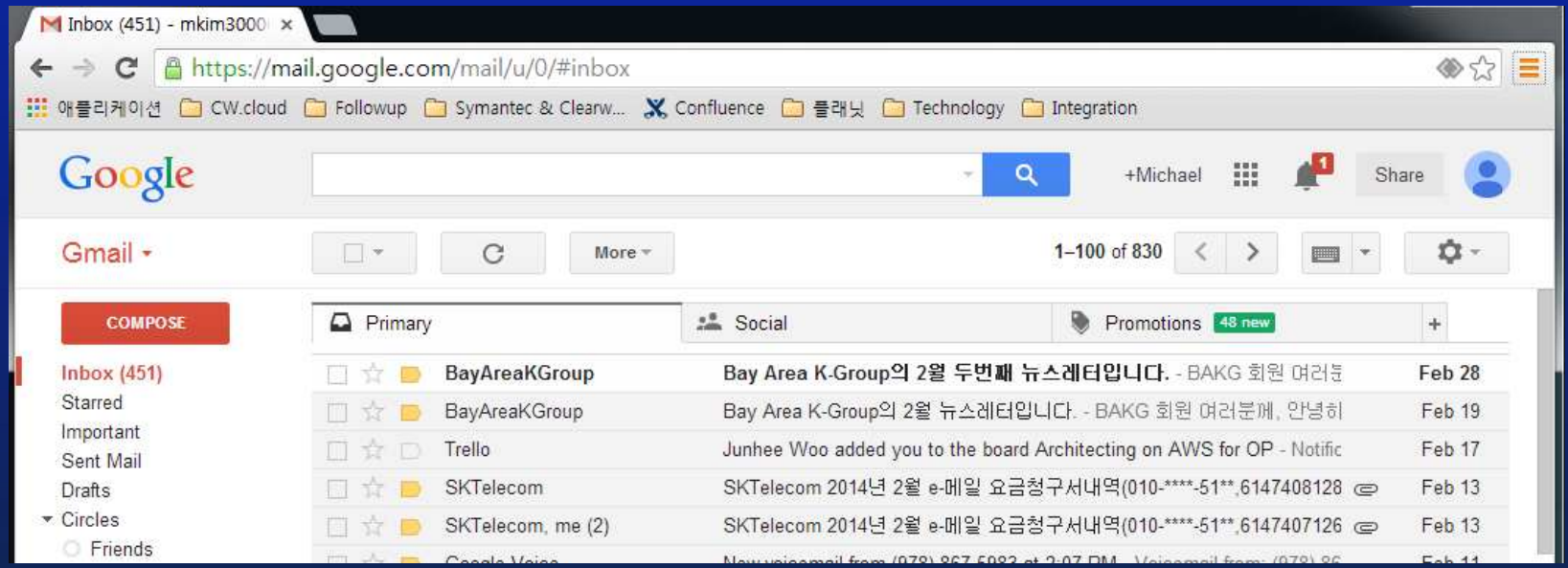
# XML vs. JSON




http://www.google.com/trends

# RESTful style - easier consumption

Gmail 내용을 어딘가에 삽입하고 싶다 ?




http://mail.google.com/mail/feed/atom

# 통상적인 proprietary API 를 사용하려면 ...

1. Native format (Windows, Linux, Corba, RPC, DCOM ...)
2. Platform dependency ... Java, C#, C++
3. Compatibility issues ... versioning & on-going update
4. Security concerns, i.e. Firewall port
5. Internal process i.e. H/W purchase & approval
6. Hosting / maintenance
7. Plumbing code development

# 가요 top 10 순위 가져오기

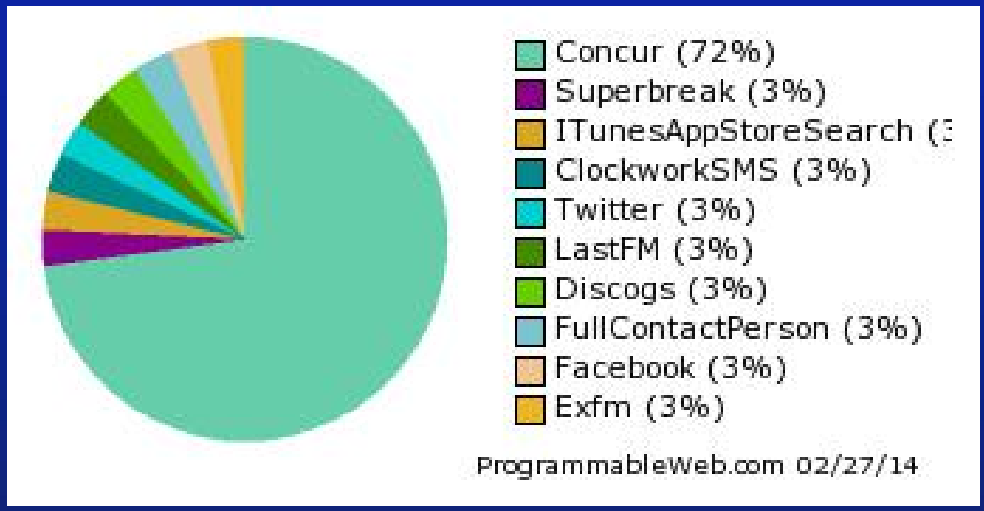
1. 내가 원하는 **API** 를 검색

   예) “*멜론 실시간 차트* ***API***”

2. **URL**을 **copy**해서 브라우저나 프로그램 내에서 호출

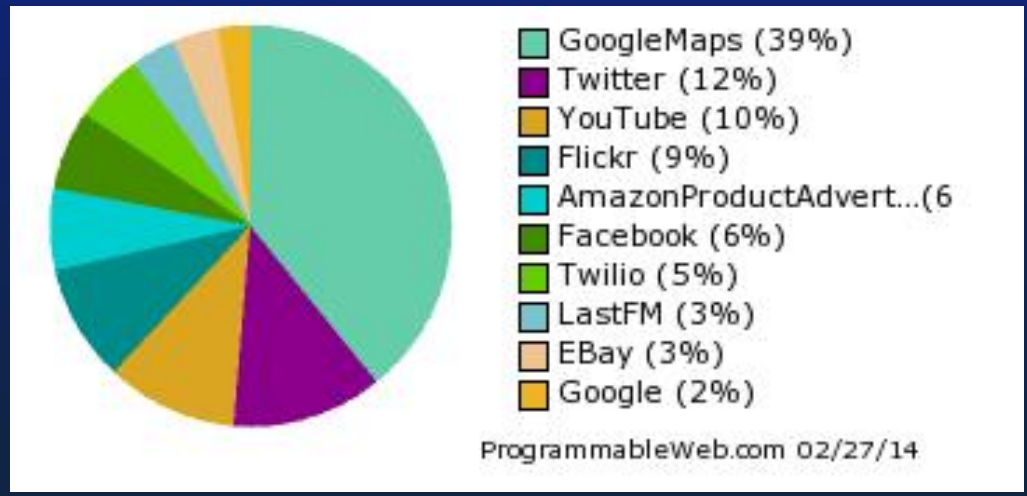
3. (선택적) **appkey**를 붙여준다.

# Global top APIs for Mashups




Last 14 days

All time



# 국내 Major players

NAVER

?

DAUM

3억건

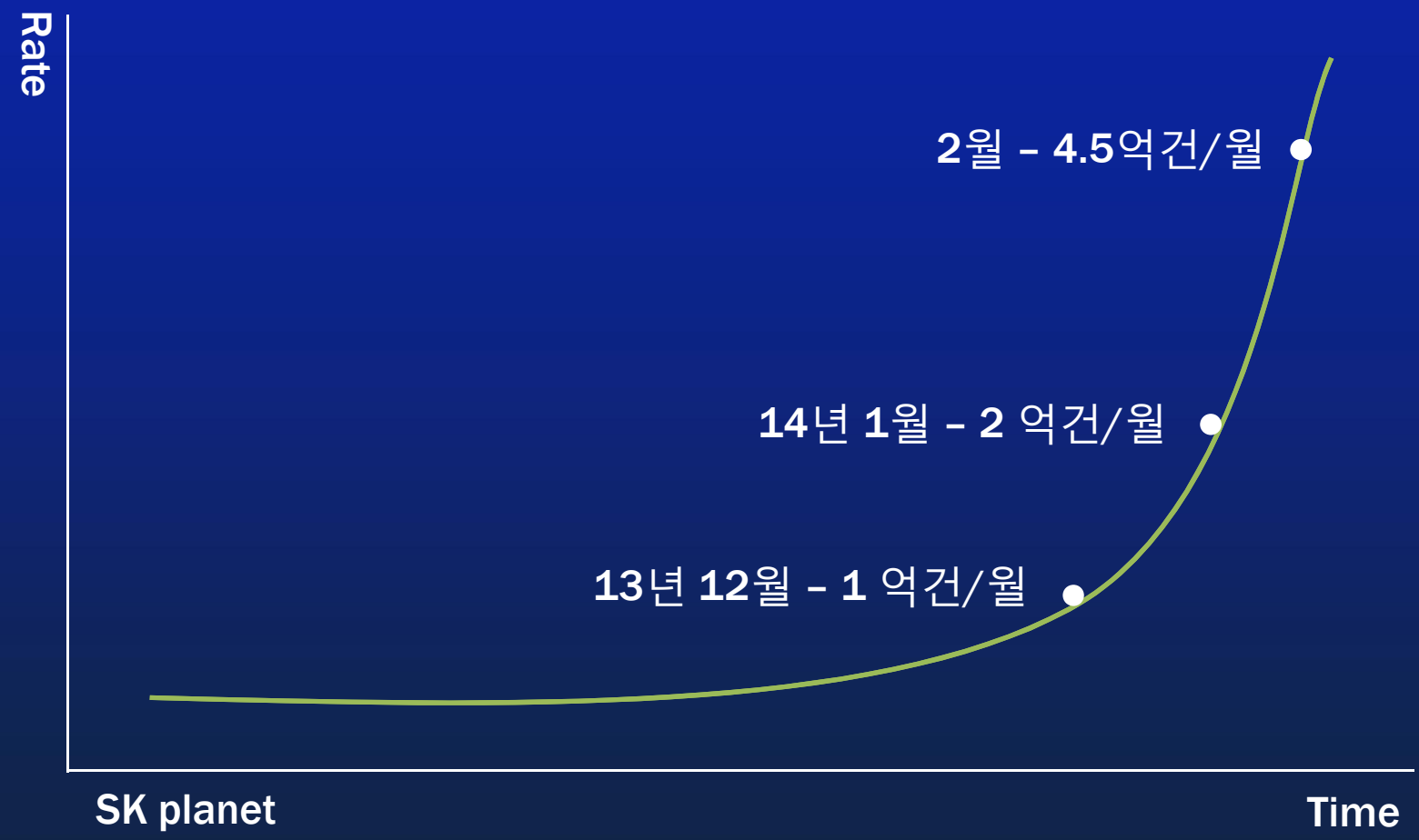
(2013.12)

SK planet

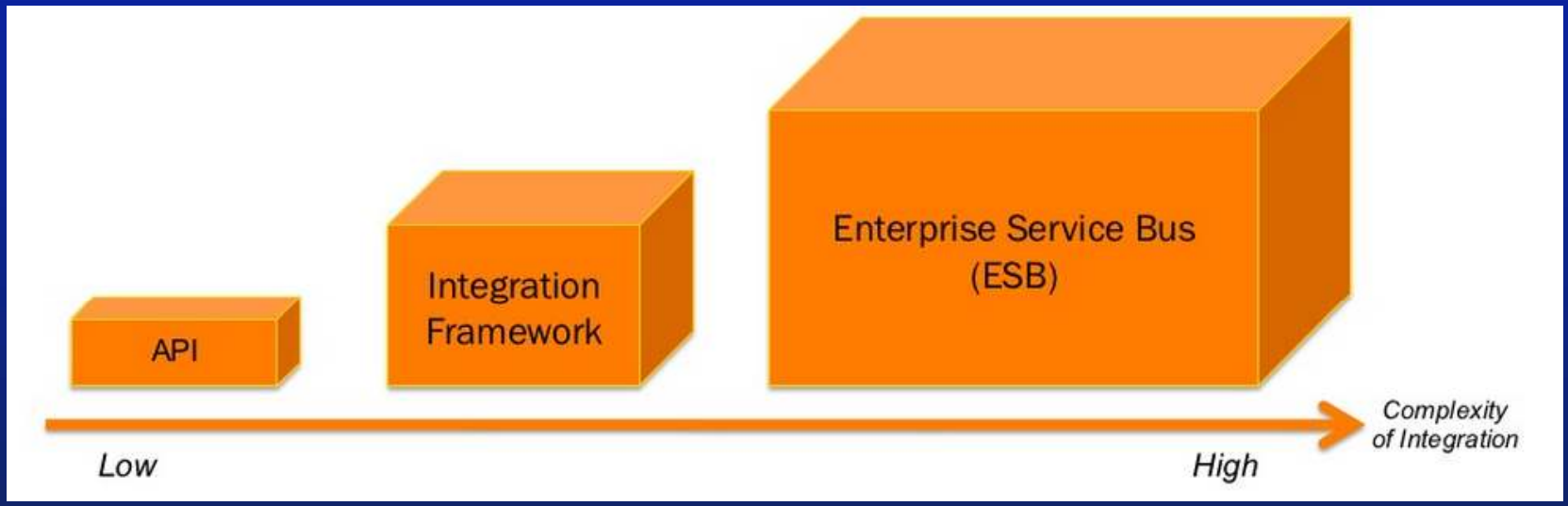
4.5 억건

(2014.02)

# API request # 증가 (예시)



# Integration alternatives




http://www.maibornwolff.de/

Proprietary
SOAP
REST

Connectivity
Routing
Transformation
ex) Spring, Apache Camel, Nintegrate

Middelware
BPM
Registry / Repository
Rules Engine
ex) Apache ServiceMix, Talend ESB, NServiceBus, IBM msg. broker, Windows Azure Service Bus , Mule,

# Strategy

- More Channels, More revenues
- Expose legacy application and data as APIs
- Brand building
- Reduce cost, Increase App development speed
  - Standard for application integration
  - Self-service / Users empowerment
  - Time-to-market
  - Reduce man-power to support partners

# More channels (판매경로), More revenue

Walgreens

미국내 최대 약국 체인 (**$75B in 2013**)

- 시스템 기능을 **API**화 시킨후 약품 판매 **6**배 증가
- 대부분 **내부** 공개 **API** 를 통해 다양한 **Channel** 개발
- 다양한 내부 시스템 개발이라면 **API**화 이전에는 왜 불가능했을까**?**

# More channels, More revenue


- 33% of peak internet downstream traffic in North America
- 1000+ 개의 Device를 지원
- 95%가 넘는 내부 API 사용량 (REST API를 이용한 인터페이스 표준화 One-Size-Fits-All OSFA 전략 – 이후 전략 변경)
- 회사 내부의 개발 문화의 변화

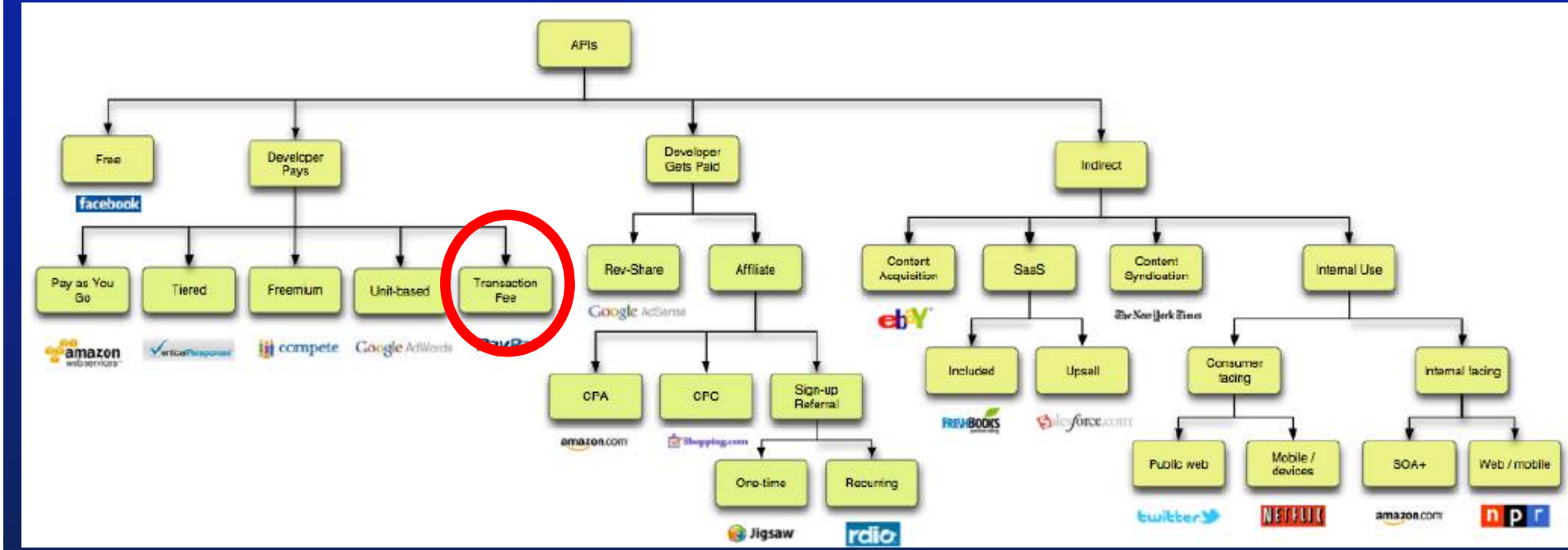

- POI/경로안내/교통정보/최근길/즐겨찾기 등 42개의 Open API 제공
- 230개 이상의 3rd Party App에서 T map API 사용
- B2B API 제공 계약을 통한 매출

# Brand Building

- Ecosystem
- Lock-in
- Brand “Powered by …”

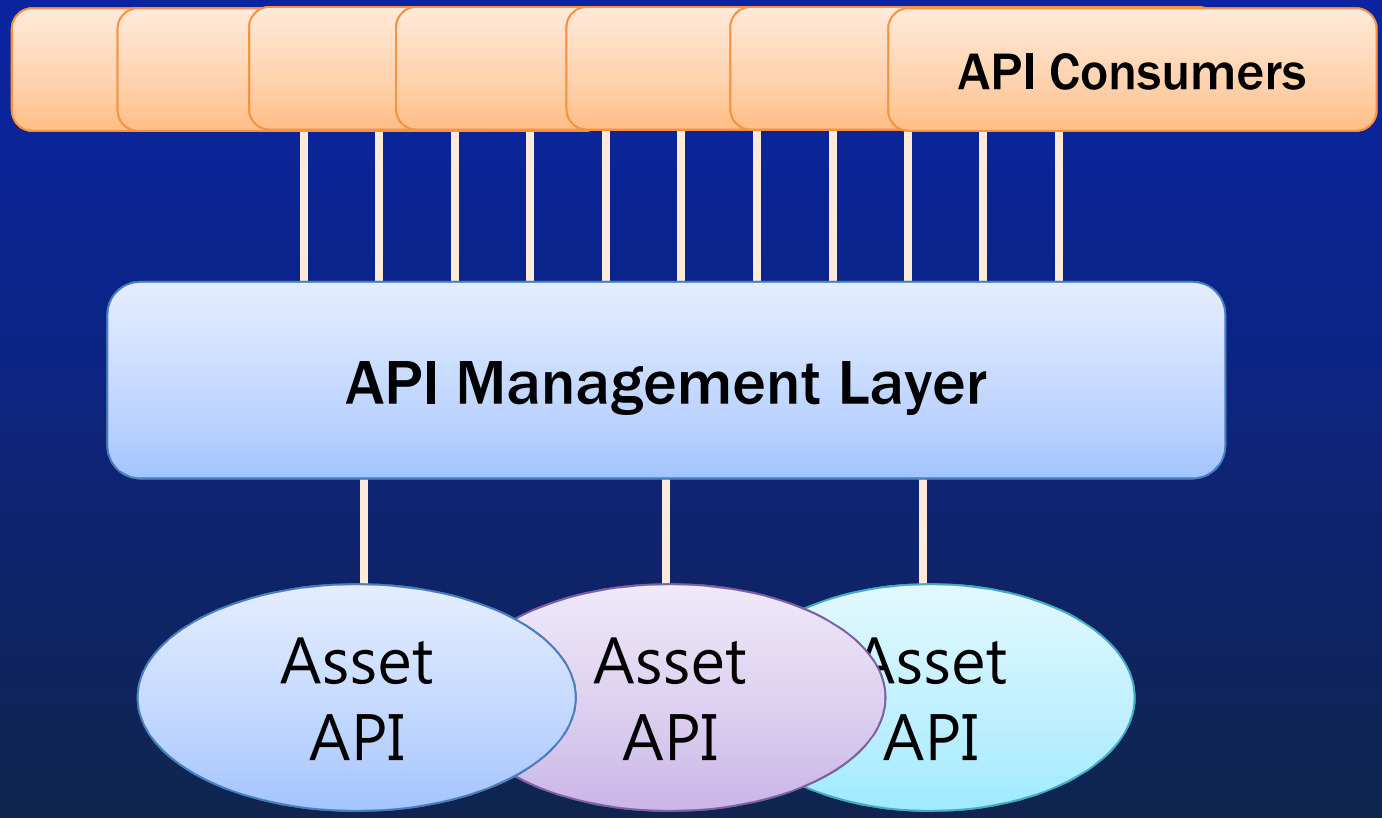

- facebook vs. myspace
  - myspace가 원래는 facebook보다 훨씬 컸습니다.
  - myspace는 현재 영향력 거의 상실
  - facebook 이 Open API 를 1년 이상 빨리 출시
  - Open API를 통해 트래픽을 2배이상 성장 시키며 추월 시작

# API business model




**Many more BMs than Transaction fee !**
**API is more of infrastructure than BM itself !**

# API management product



# API management product type

- Appliance based gateway
- Cloud-based Proxy (SaaS)
- Plug-In
- Install-based suite
- Custom built

# Global API management suites


Acquired by Intel


IPO ?


Acquired by CA

# 통상적인 API management layer의 features

- API Services
  - Gateway
  - Policy / Security
  - Branding
  - Identity
  - Cache
  - Throttling
- Developers Services
  - Web presence
  - Documentation
  - Developers community
- Analytics Services

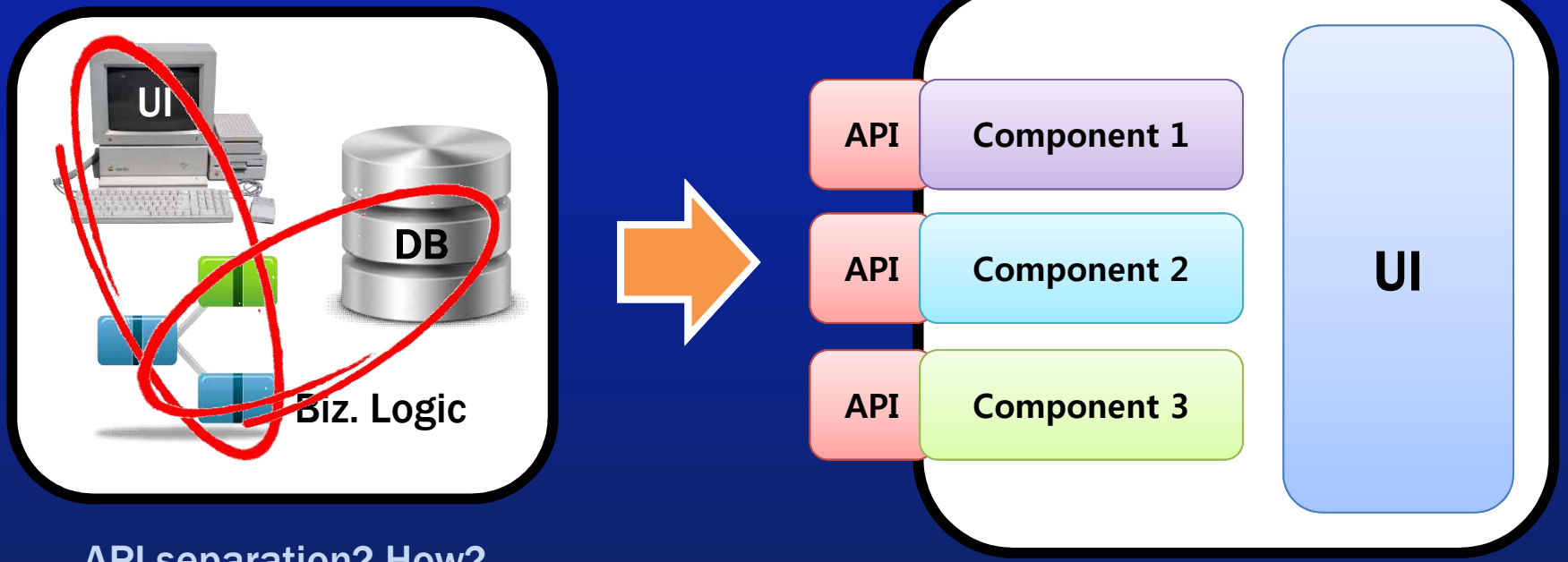
# 통상적인 시스템의 Paradigm 변화

# 통상적인 시스템의 Paradigm 변화

# 통상적인 시스템의 Paradigm 변화

# 통상적인 시스템의 Paradigm 변화

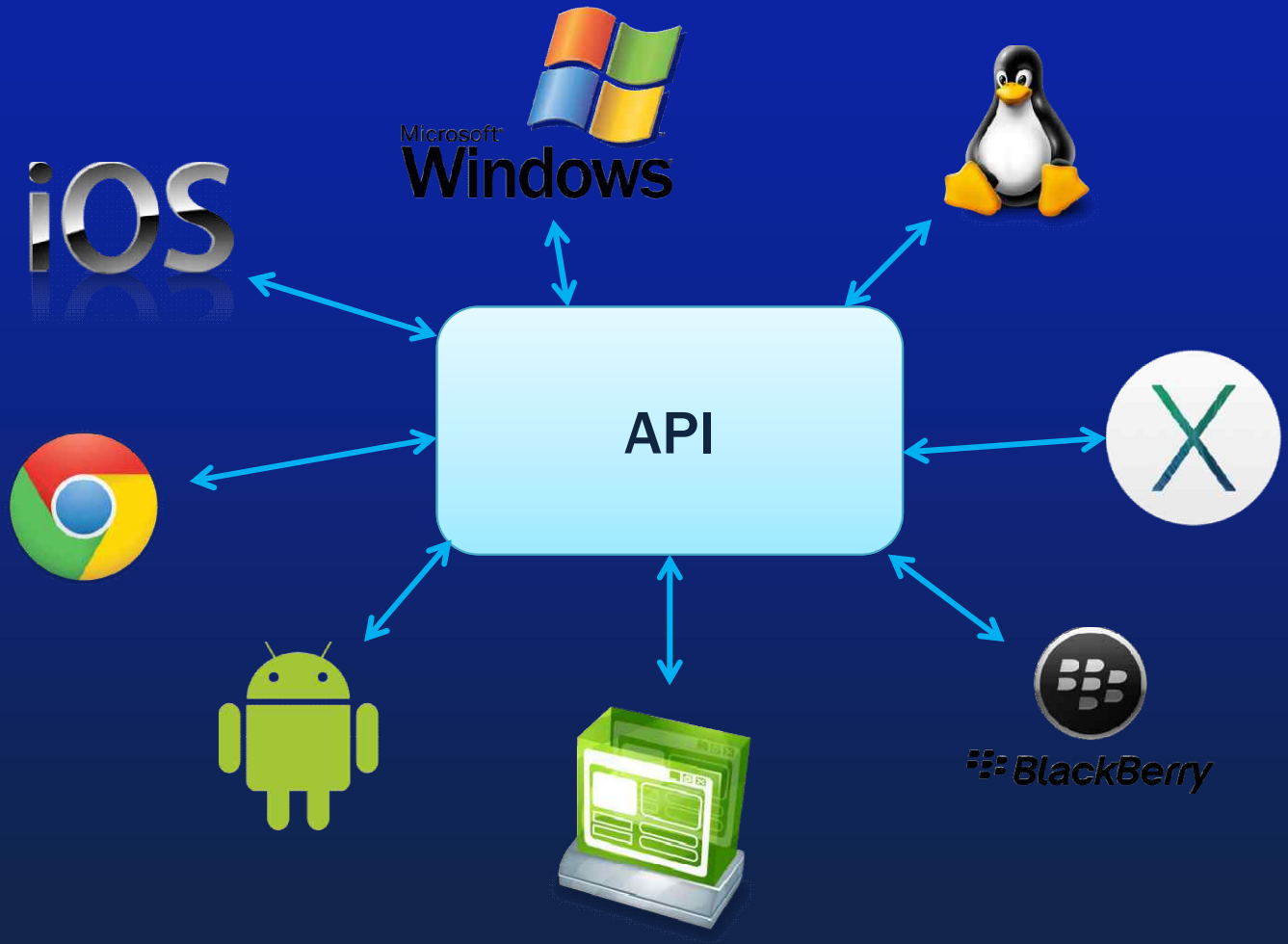
# API centric design




API separation? How?

- Component / API를 바탕으로 한 UI 를 개발
- Dog fooding 을 통한 API의 사용성 검증
- API를 통해 동일한 기능의 앱을 제 3자가 개발할 수 있는 수준이 이상적인 Open API 의 제공의 형태

# API Centric Design



# Multi-tenancy / Multi-sites


- 전통적인 개념의 web site와 Open API 개발자 센터가 Merge되는 경향
- 개별 product 혹은 product group별로 Grouping되어 고유의 management / web presence
- 각 grouping 별로 self-contained 된 User experience 제공 (예: 게임 커뮤니티)

# BaaS – Backend as a service

- Save date in the cloud
- Push service
- Analytics
- Make app social (ie. OAuth)
- Run batch job in cloud
- Dashboard / administration
- Massive scale
- Share data between platforms
- ...

- 모든 것이 **REST API** 혹은 그것을 **wrapping** 한 **SDK** 를 통해서 이루어진다.

# Digital 플랫폼의 새로운 기준

- Mobile first
- Exposure centric (app/web presence)
- Self serve (start w/ simple email confirmation)
- Web scale (Elasticity)
- Carrier grade (Reliable)

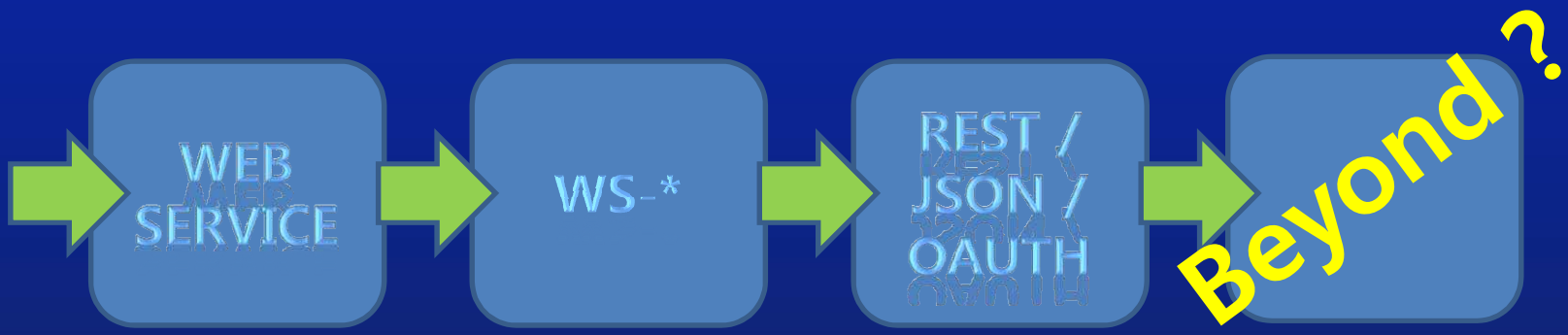
By Apigee CEO, Chet Kapoor

➔ Open API

# API 와 management layer 에 대한 common myths

- Open API – 당장의 ROI 없이 외부에 오픈해야 합니까?
  - 내부 / 파트너 / 외부 가 모두 대상이 될 수 있고, 내부 오픈을 통해 시스템 통합의 효과
  - Public or private API : Private API still requires login!
- API management 시스템이 overhead를 가져오지 않습니까?
  - 통상 ms 단위의 relay시간이 필요하며 전체 수초 단위의 end-to-end API response 에 비하면 아주 적은 비용
- API management 시스템을 공유하는 다른 서비스로부터 영향을 받습니까?
  - API 혹은 User 별로 설정된 Throttling을 통해 도리어 DDoS 나 Slashdot effect 같은 예상치 못한 트래픽으로부터 보호
- 시스템의 처리용량이 제한 받지 않습니까?
  - 통상적인 캐시 레이어의 사용으로 도리어 Asset layer의 응답 속도와 처리용량이 증가

# Beyond REST / JSON




- SOA
- Firewall friendly HTTP/S
- SOAP / XML / WSDL

- Snappy !
- Web 2.0

- Security
- Transaction

# Beyond REST / JSON – problem statement

- REST 가 간결하나 기능적 한계
- Connection / stateless HTTP
  Data oriented API vs. content oriented API
- Interface definition

# Beyond REST / JSON – New players

- Protocol buffer (Google)
  - Binary
  - Protocol in IO|
  - IDL to specific code
- Apache Thrift
  - Google protocol buffer (TCP/binary) + α
- Apache Avro
  - No code generation
  - Better versioning
- Microsoft WCF




# Alice in wonderland

# Fast follower vs. Market leader

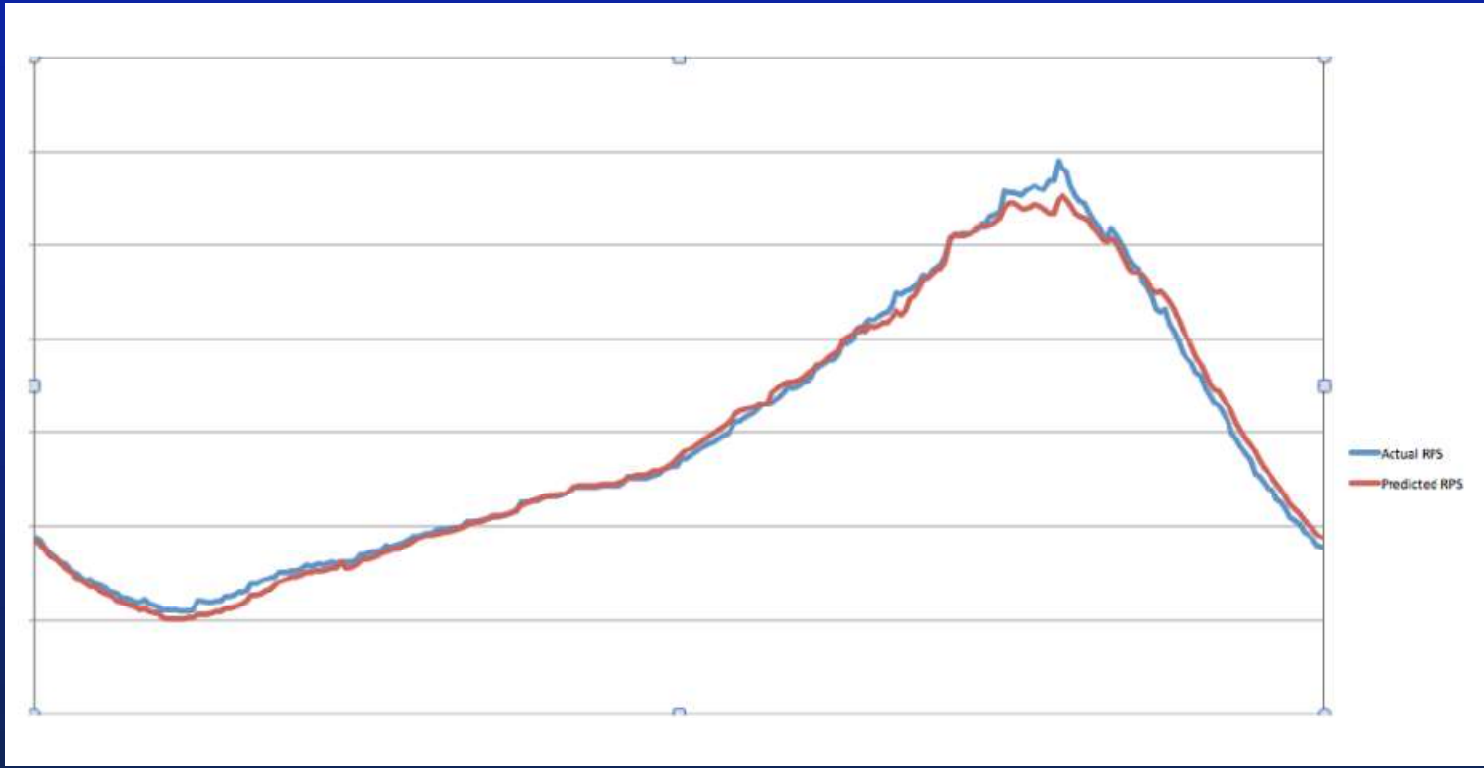
- 창조에 집중 그러면 일상적인 일은 ?
- **Fast follower**는 시간이 없다? **Market leader**는 창조가 없다?
  - 창조를 하는 동안 시간을 아껴주는 시스템 필요
- 나도 다 할 수 있다는 생각 **vs. Don't invent the wheels.**
  - 글로벌 기업에서 자체 경쟁력 확보를 위한 이유가 아닌 이유로 이미 존재하는 동일한 컴포넌트를 개발하는 것은 죄악시
    - 될 수 있으면 남의 손을 빌려서 이루라 !
    - 능력이 아니라 시간의 문제
- 예) 3 개월 만에 만들어지는 1000 페이지 책의 비밀

# Open API 전략의 시사점

- Front-end 위주의 소모적인 경쟁을 회피하는 대신 Back-end API 배포를 통한 Lock-in 및 Eco system 구축
- Global platform 성격의 프로덕트 에서 Open API는 기본 기능으로 인식
- 자유로운 Integration을 통한 Multi-channel 확보
- 플랫폼과 Standalone system의 차이는 Open API; 어떤 프로덕 이던지 플랫폼으로 진화가 가능하다 !

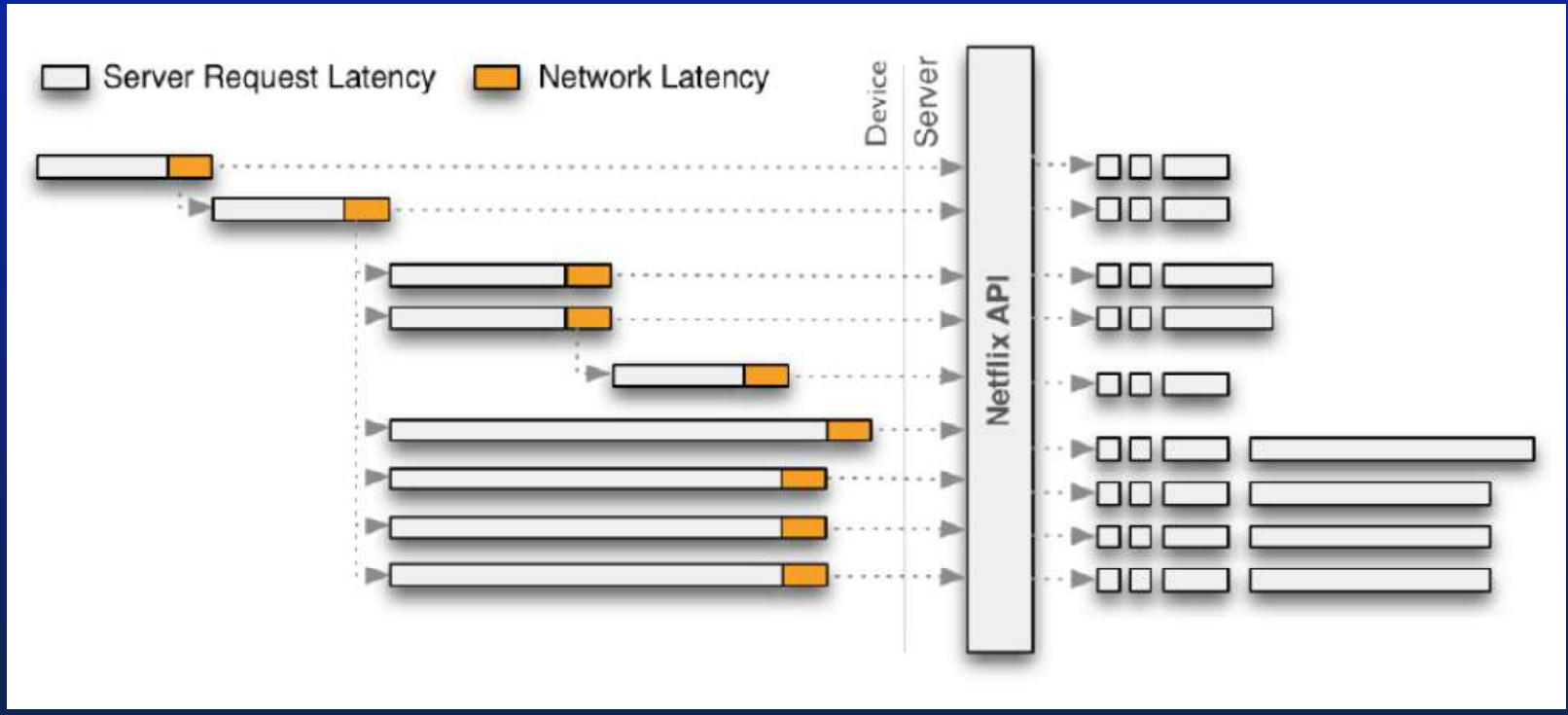
# Thank you

# Predictive + Reactive auto scaling



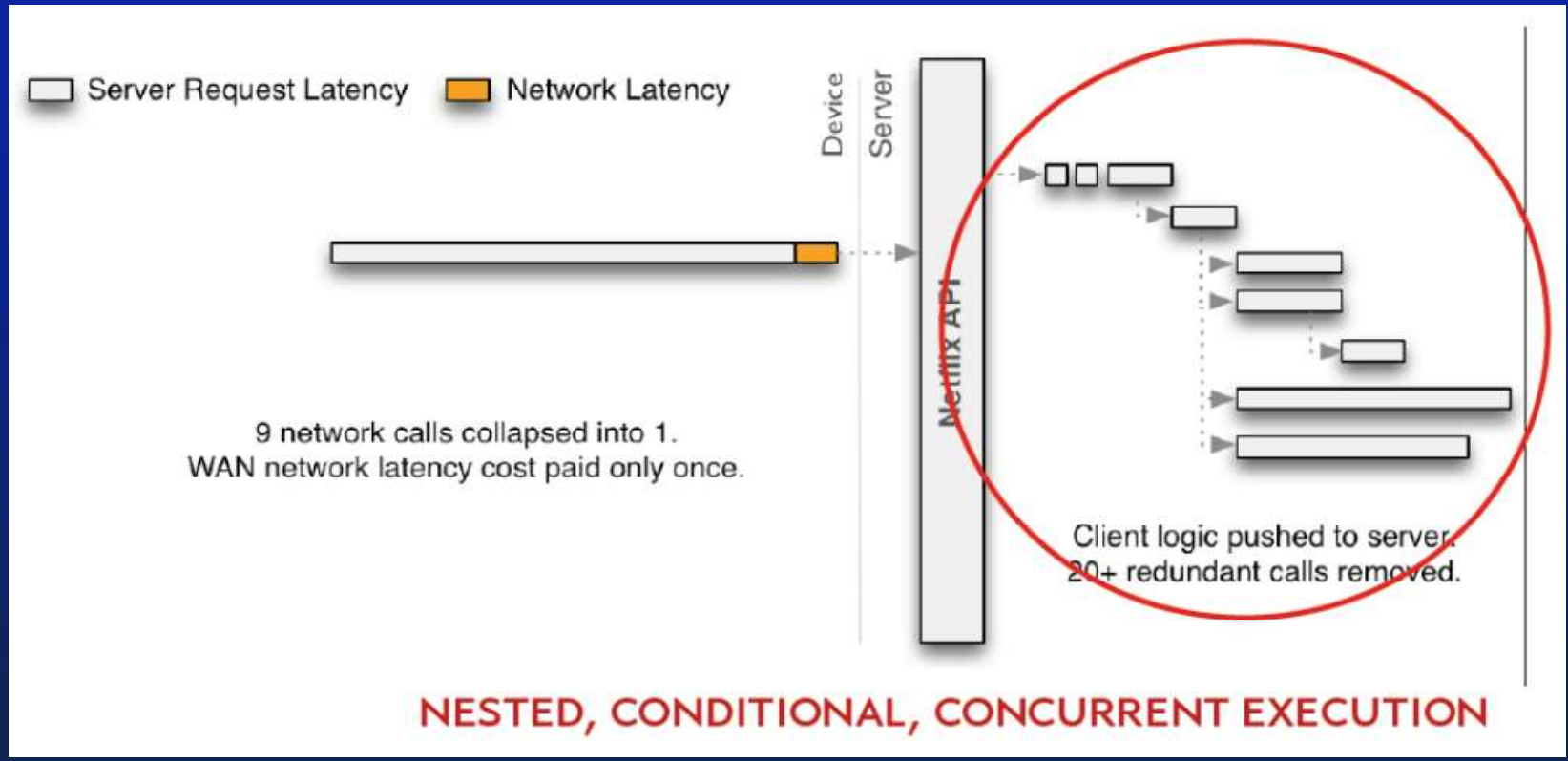

Blog.netflix.com

# Call pattern re-architecture - Before




Blog.netflix.com

# Call pattern re-architecture - After




Blog.netflix.com