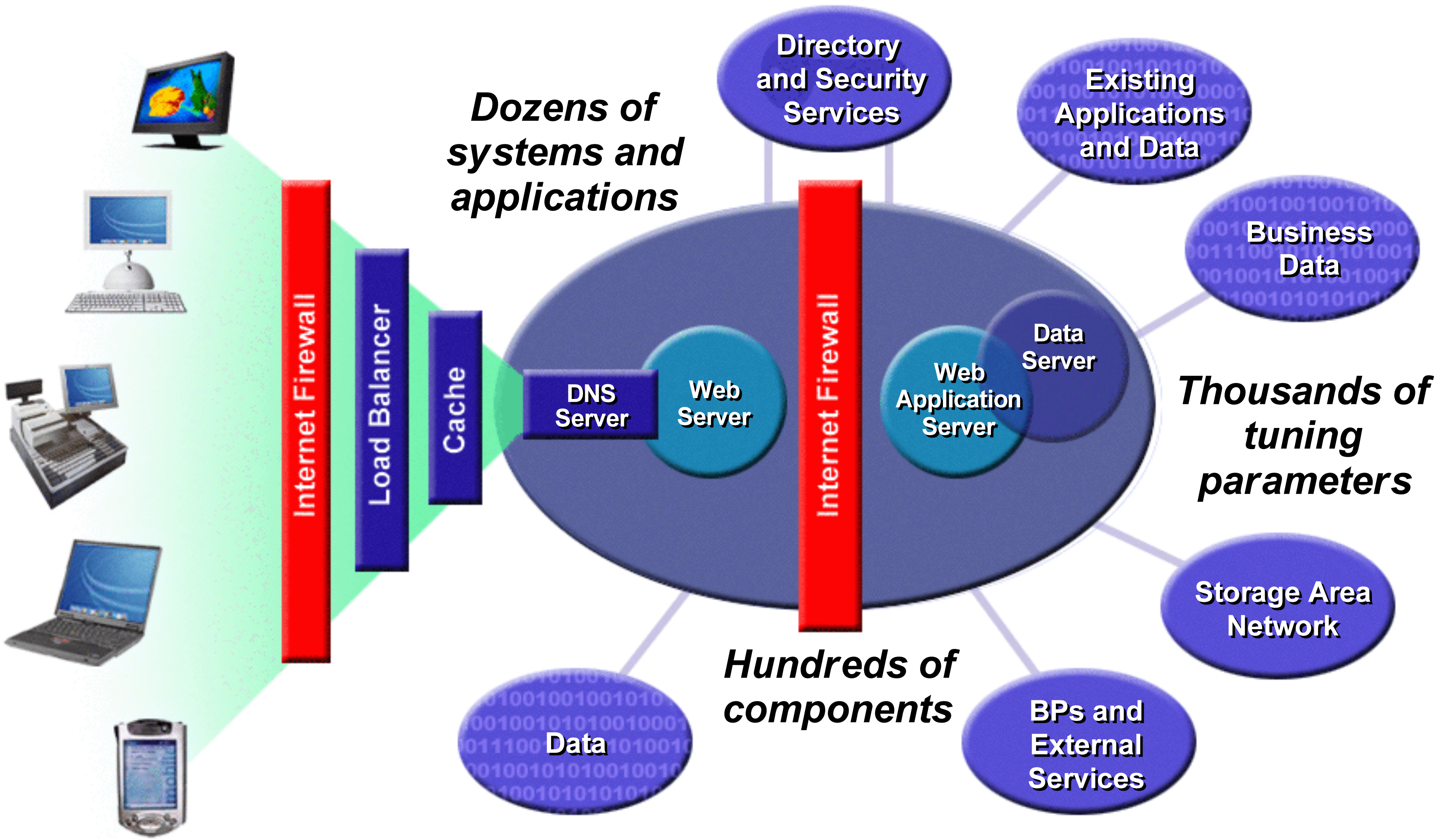
Autonomic Computing

Complex Heterogeneous Systems

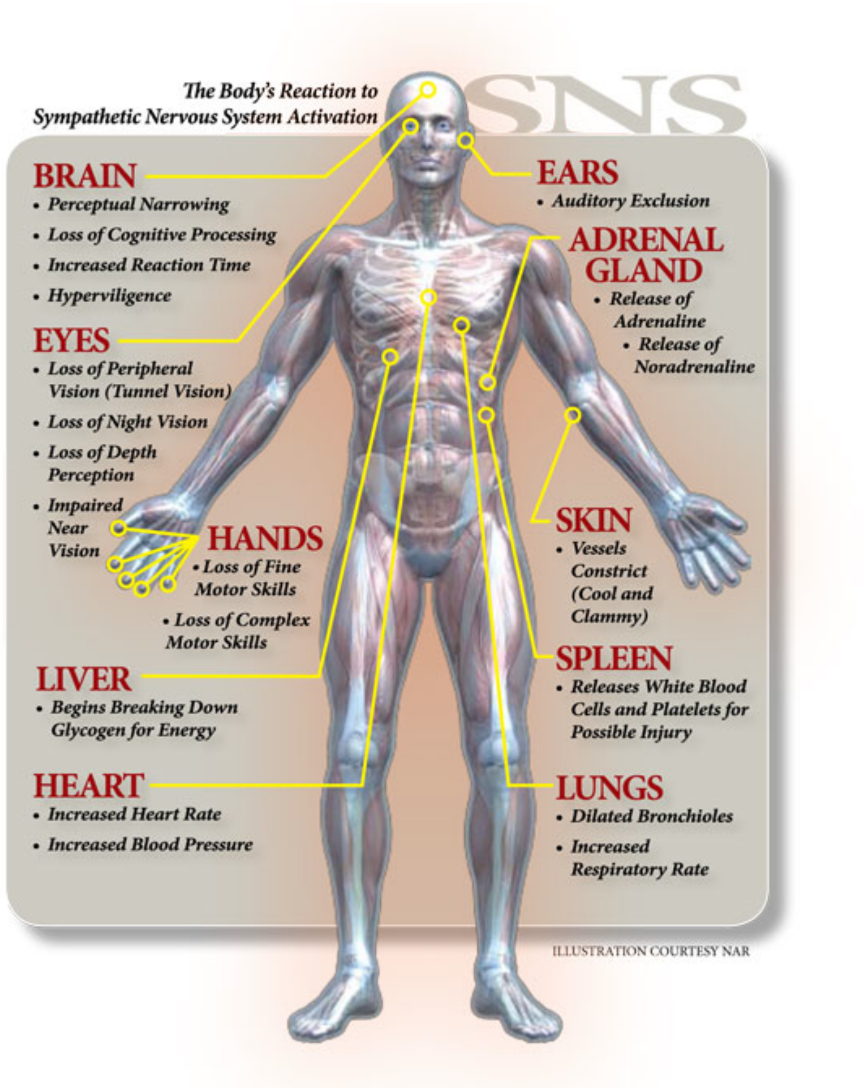


Problems

- Administration of individual systems is increasingly difficult
  - Thousands of configuration, tuning parameters
- Heterogeneous systems are becoming increasingly connected
  - Integration becoming ever more difficult
- Designers can't intricately plan interactions among components
  - Increasingly dynamic; more frequently with unanticipated components
- More of the burden must be assumed at run time
  - But human system administrators can't assume the burden; already
    - 6:1 cost ratio between storage admin and storage
    - 40% outages due to operator error
- We need self-managing computing systems
  - Behavior specified by sys admins via high-level policies
  - System and its components figure out how to carry out policies

Autonomic Option

- **Autonomic computing**
  - Named after autonomic nervous system
  - Systems can manage themselves according to an administrator's goals
  - Self-governing operation of the entire system, not just parts of it
  - New components integrate as effortlessly as a new cell establishes itself in the body



Digital Preservation Scenario

- A system providing permanent access to some PDF digital content
    - Continual check for new Acrobat Reader versions
    - Download and install new Acrobat packages
    - Automatic verification of digital signatures
    - Replicate everything if hot spot
    - Revert to older version if errors detected
    - Hardware failures

Self Configuration

- Today
  - Multi-vendor, multi-platform systems must be installed, configured and integrated through a time-consuming, error-prone, human-controlled procedure
- Future
  - Automated configuration of components according to high-level policies
  - Dynamic adaptation to changing environment
  - Addition of new features dynamically

Self Optimization

- Today
  - Thousands of configuration, tuning parameters and new parameters with new releases
- Future
  - Reallocating resources to improve overall utilization or to ensure that particular business transactions can be completed in a timely fashion
  - Monitor and tune resource utilisation
  - Dynamic partitioning, workload management

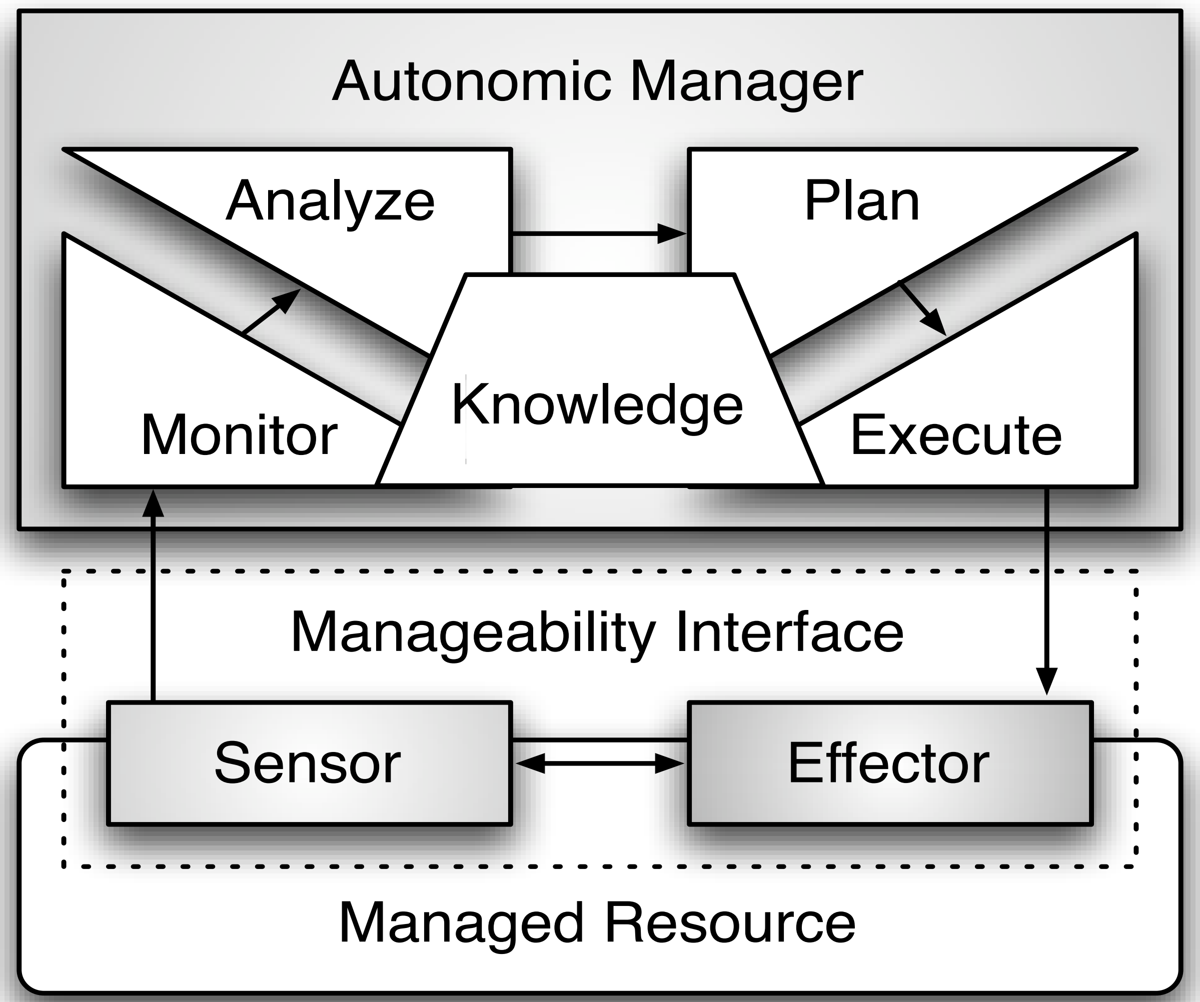
Self Protection

- Today
  - Manual detection and recovery from attacks and cascading failures.
- Future
  - Anticipate/Identify, detect and protect from attacks
  - Extend existing security infrastructure to achieve this

Self Healing

- Today
  - Problem determination in large, complex systems can take a team of programmers working for weeks
- Future
  - Discover, diagnose and react to disruptions
  - Handling failure and isolating a component

Architecture



Architectural Consideration

- Autonomic elements will function at many levels
  - At the lower levels
    - Limited range of internal behaviors
    - Hard-coded behaviors
  - At the higher levels
    - Increased dynamism and flexibility
    - Goal-oriented behaviors
- Autonomic elements will manage
  - Internal behavior
  - Relationships with other autonomic elements
- Individual component level
  - Make each component more intelligent
  - Provide support infrastructure around this intelligent component
- Interaction level
  - Facilitate better interaction between components in some way
  - Allow "useful" interactions to "emerge"

Engineering Challenges (I)

- Design, test and verification
- Lifecycle
- Upgrading
- Monitoring mechanisms
- Adaptation mechanisms
- Knowledge aggregation/distribution
- Information Filtering
- Interaction Specification
- Interaction Implementation
- Negotiation
- Hierarchical management

Engineering Challenges (II)

- Systemwide issues
  - Authentication, encryption, signing
  - Introspection/Intercession
  - Robustness against attacks
- Goal specification
  - Humans-provided goals and constraints
  - Policies
  - Protection from input goals that are inconsistent, implausible, dangerous, or unrealizable

Scientific Challenges

- Behavioral abstractions and models
  - Mapping from local behavior to global behavior
- Robustness and reliability
  - Will work even with errors in design parameters
  - Several mechanisms cooperating towards the same goal
- Learning and optimization
  - ACs continually adapt to their environment that consists of other ACs
  - There are no guarantees of convergence
- Negotiation theory

Autonomic control of the Apache Web Server

J. Hellerstein et al.
IBM Thomas J Watson Research Center

Metrics

- Master process + pool of worker processes
- Each worker process handles interaction with a Client
- Worker processes limited by MaxClients
- Worker Process: idle, waiting and busy
  - Idle (no TCP connection made)
  - Waiting (waiting for HTTP request from client)
  - Busy (processing request)
- Persistent HTTP/1.1: TCP connection remains open between consecutive HTTP requests (reduces time to set up a connection)
- Persistent connection can be terminated by master or client process – if waiting time exceeds max. allowed by KeepAlive

Workload and objective

- To maintain CPU and Memory criteria, it is necessary to tune manually
- Achieved by adjusting MaxClients and KeepAlive parameters
- Dynamic workload (generally unpredictable) requires continuous re-tuning
- Trying to follow changes resulting from dynamic workload can be continuous process

Apache Web Server Modelling

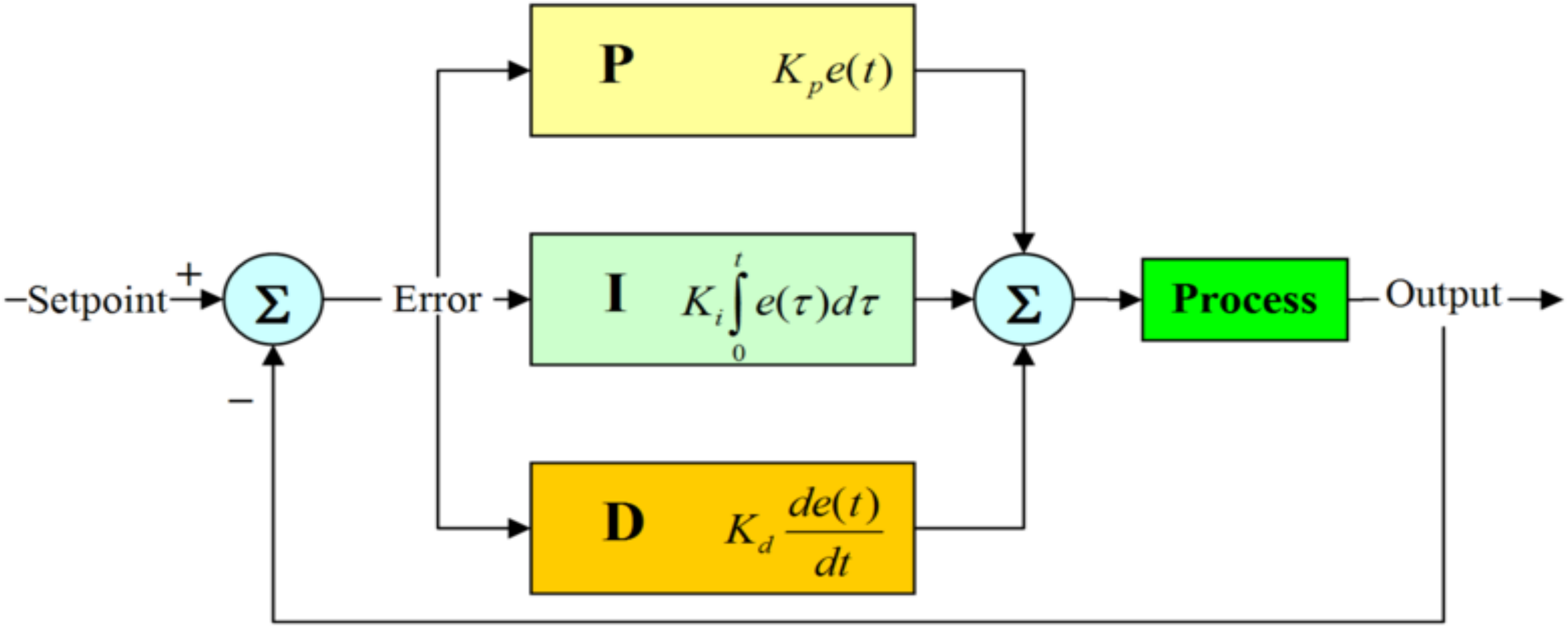
- Build a mathematical model of the system
  - Queuing theory
  - Data analysis based
- Mathematical model
  - Requires understanding of inner workings of server
  - May need to know about particular properties (exceptions) of the way the server operates
- Data-based model ("blackbox" approach)
  - Gather data of system in the "wild"
  - Assume have covered sufficient number of test cases
- User Input
  - Range of Tuning Parameters: MaxClient [1,1024]; KeepAlive [1,50]
  - Max delay required for tuning parameters to take effect on the performance metrics: MaxClients (10m); KeepAlive (20m)

$$\begin{bmatrix} \mathrm{CPU}_{k+1} \\ \mathrm{MEM}_{k+1} \end{bmatrix} = A \cdot \begin{bmatrix} \mathrm{CPU}_{k} \\ \mathrm{MEM}_{k} \end{bmatrix} + B \cdot \begin{bmatrix} \mathrm{KeepAlive}_{k} \\ \mathrm{MaxClients}_{k} \end{bmatrix}$$

Feedback control

- PID (proportional-integral-derivative) control
- Correct error between a measured process variable and a desired point
- Calculating and outputting a corrective action to adjust process accordingly
- Proportional: reaction to current error
- Integral: reaction based on recent error (time based)
- Derivative: reaction based on rate by which error has been changing
- Use a weighted sum of the three modes
- Output as a corrective action to a control element

PDI Controller



$$\text{Output}(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau + K_d \frac{de}{dt}$$

UNIVERSITÀ DI PISA

Results