Fundamentals of distributed systems

Alisson Sol
Knowledge Engineer
Engineering Excellence
June 08, 2011

Public version

The importance for you

- Information about the current “inflection point”
  - Mature
    - Mainframe, desktop, graphical user interface, client/server…
  - **Evolving**
    - **Devices, online services, distributed processing…**
  - Emerging
    - Quantum computing, biodevices…

The old times

```
10 PRINT "HELLO WORLD"
20 END
```

Windows “Hello World”

```
#include <windows.h>

LRESULT CALLBACK WndProc (HWND, UINT, WPARAM, LPARAM) ;

int WINAPI WinMain (HINSTANCE hInstance, HINSTANCE
        hPrevInstance,
                PSTR szCmdLine, int iCmdShow)
{
    static TCHAR szAppName[] = TEXT ("Hello World") ;
    HWND         hwnd ;
    MSG          msg ;
    WNDCLASS     wndclass ;

    wndclass.style         = CS_HREDRAW | CS_VREDRAW ;
    wndclass.lpfnWndProc   = WndProc ;
    wndclass.cbClsExtra    = 0 ;
    wndclass.cbWndExtra    = 0 ;
    wndclass.hInstance     = hInstance ;
    wndclass.hIcon         = LoadIcon (NULL, IDI_APPLICATION) ;
    wndclass.hCursor       = LoadCursor (NULL, IDC_ARROW) ;
    wndclass.hbrBackground = (HBRUSH) GetStockObject
        (WHITE_BRUSH) ;
    wndclass.lpszMenuName  = NULL ;
    wndclass.lpszClassName = szAppName ;

    if (!RegisterClass (&wndclass))
    {
        MessageBox (NULL, TEXT ("This program requires Windows
        NT!"),
                  szAppName, MB_ICONERROR) ;
        return 0 ;
    }
```

```
hwnd = CreateWindow (szAppName,                  // window class name
                     TEXT ("Hooray its Hello World"), // window caption
                     WS_OVERLAPPEDWINDOW,        // window style
                     CW_USEDEFAULT,              // initial x position
                     CW_USEDEFAULT,              // initial y position
                     CW_USEDEFAULT,              // initial x size
                     CW_USEDEFAULT,              // initial y size
                     NULL,                       // parent window handle
                     NULL,                       // window menu handle
                     hInstance,                  // program instance handle
                     NULL) ;                     // creation parameters

    ShowWindow (hwnd, iCmdShow) ;
    UpdateWindow (hwnd) ;

    while (GetMessage (&msg, NULL, 0, 0))
    {
        TranslateMessage (&msg) ;
        DispatchMessage (&msg) ;
    }
    return msg.wParam ;
}

LRESULT CALLBACK WndProc (HWND hwnd, UINT message, WPARAM wParam, LPARAM
        lParam)
{
    HDC         hdc ;
    PAINTSTRUCT ps ;
    RECT        rect ;

    switch (message)
    {
```

```
case WM_PAINT:
        hdc = BeginPaint (hwnd, &ps) ;

        GetClientRect (hwnd, &rect) ;

        DrawText (hdc, TEXT ("Hello World , Windows
      style!"), -1, &rect,
                DT_SINGLELINE | DT_CENTER |
      DT_VCENTER) ;
        EndPaint (hwnd, &ps) ;
        return 0 ;

    case WM_DESTROY:
        PostQuitMessage (0) ;
        return 0 ;
    }
    return DefWindowProc (hwnd, message, wParam,
      lParam) ;
}
```

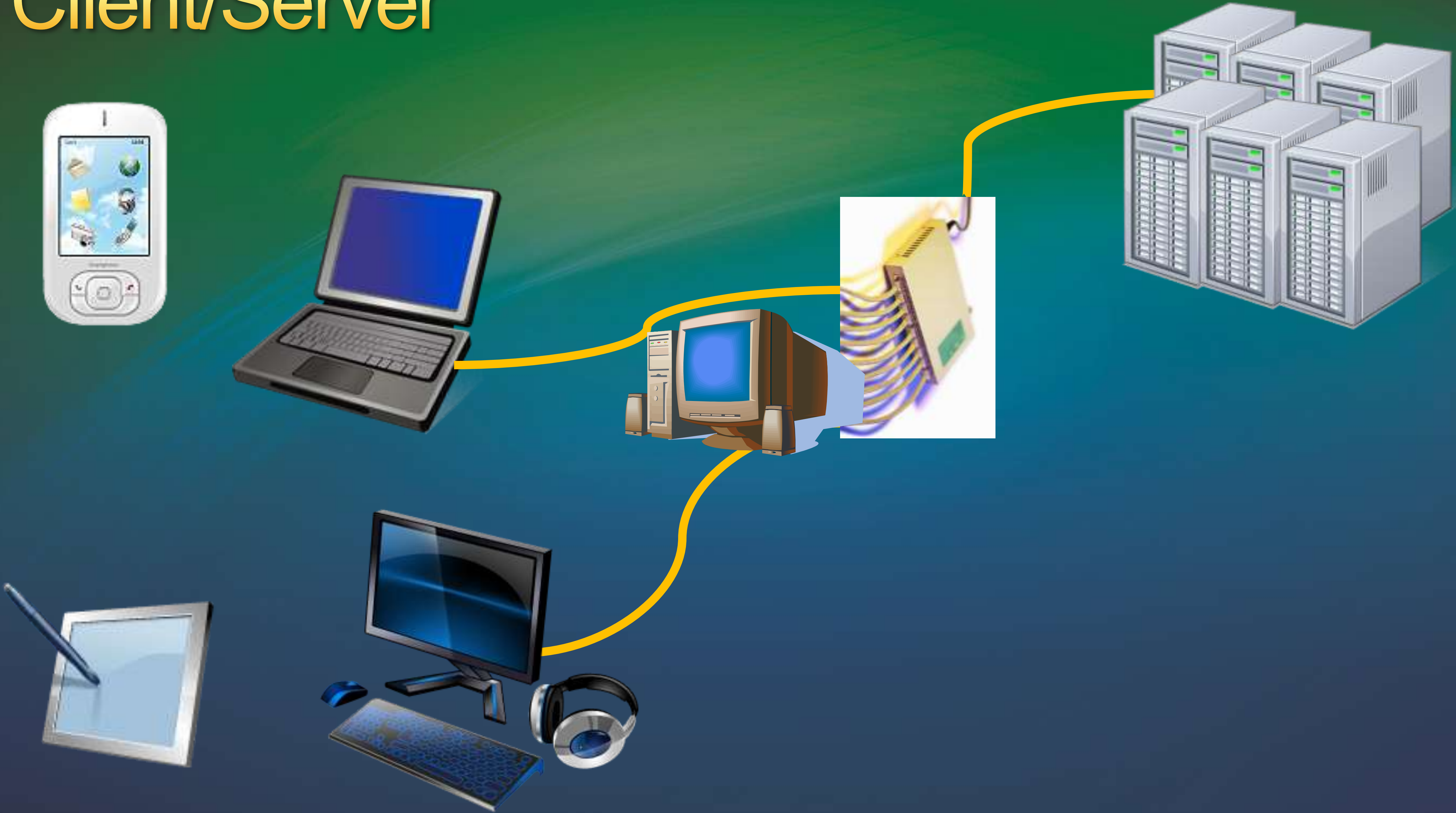
C# “Hello World”

```
using System;

namespace HelloNameSpace
{
    public class HelloWorld
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

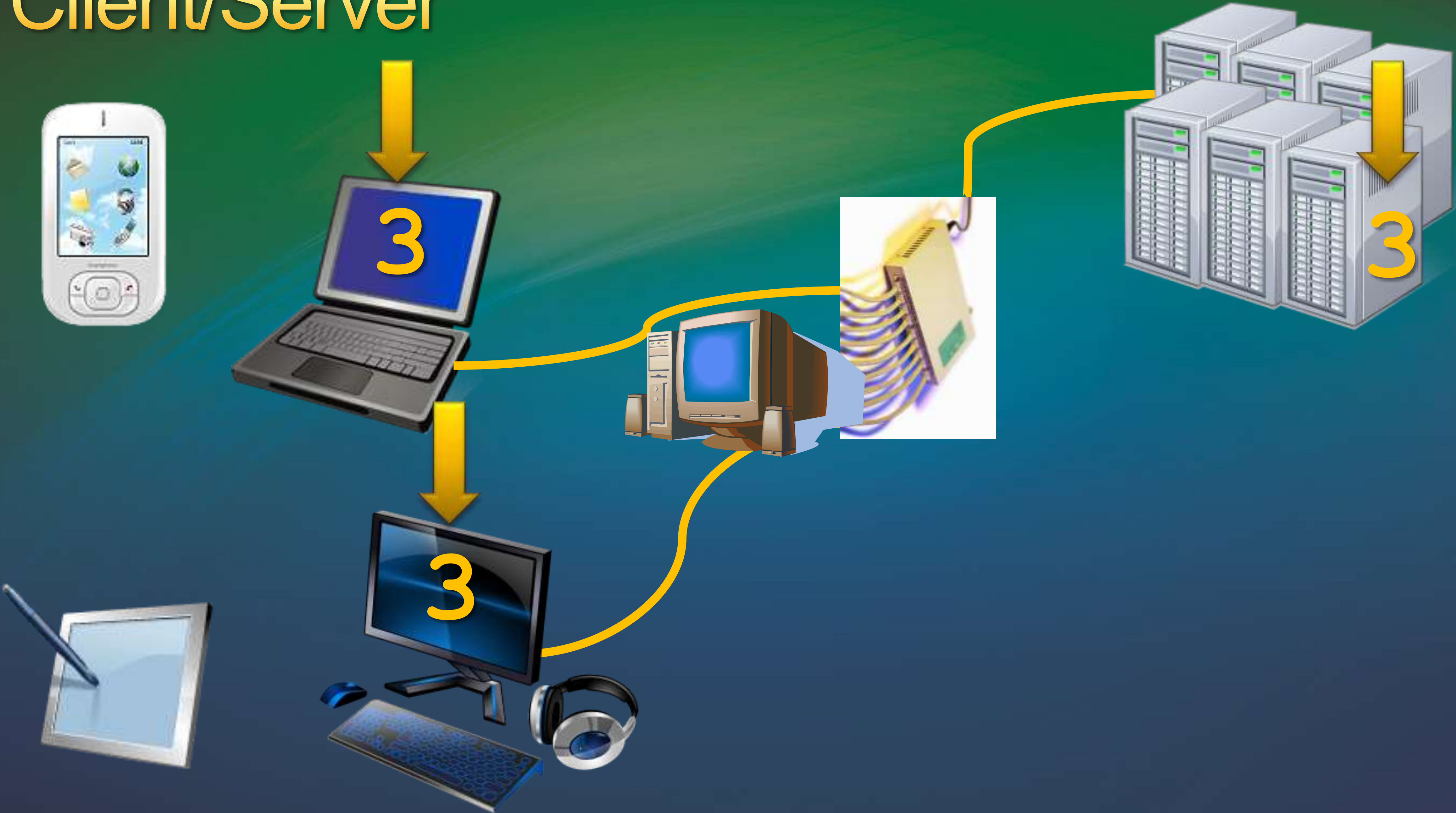
Mature programming technologies

- Mainframe, desktop, graphical user interface, client/server

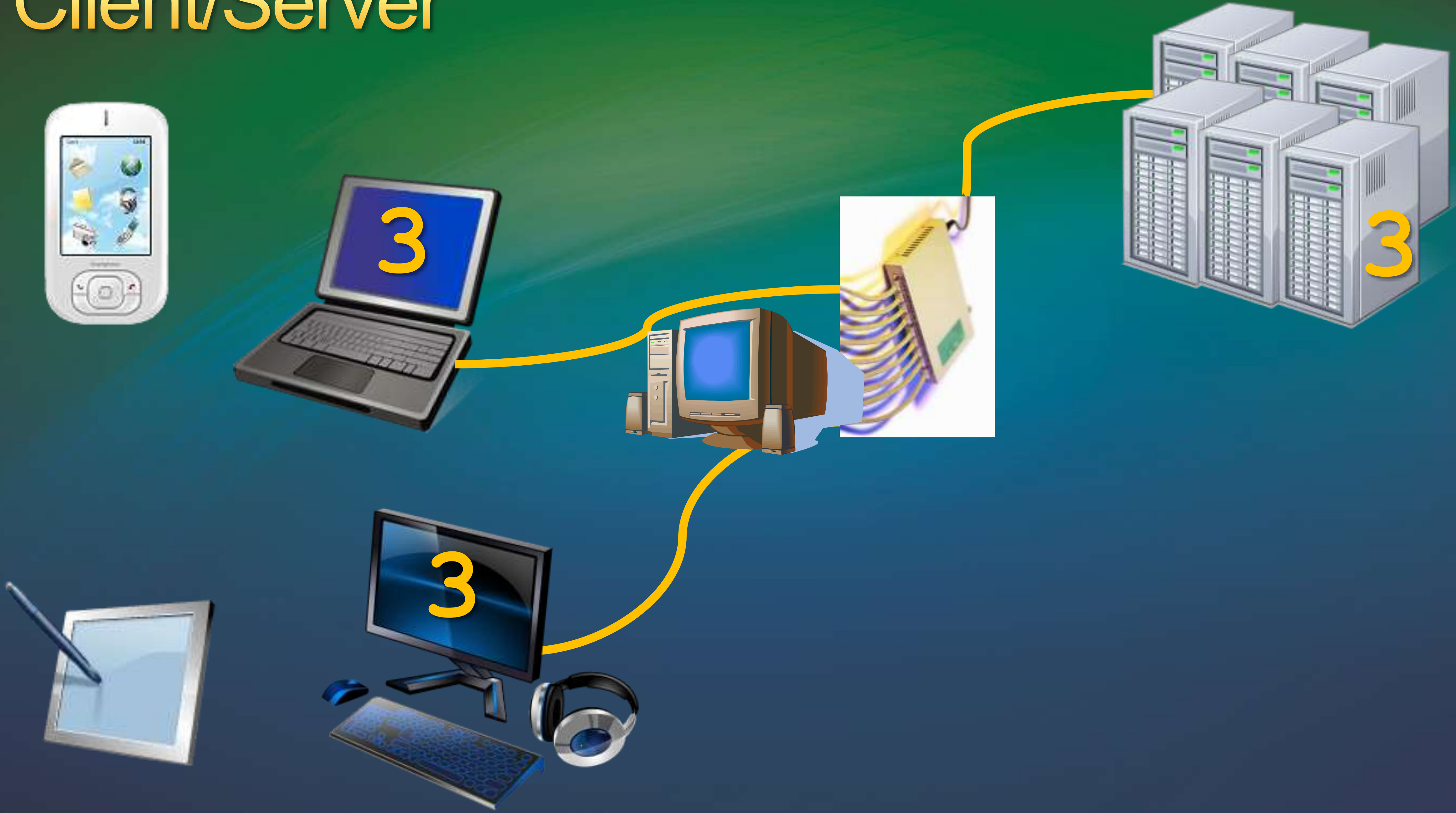
Client/Server



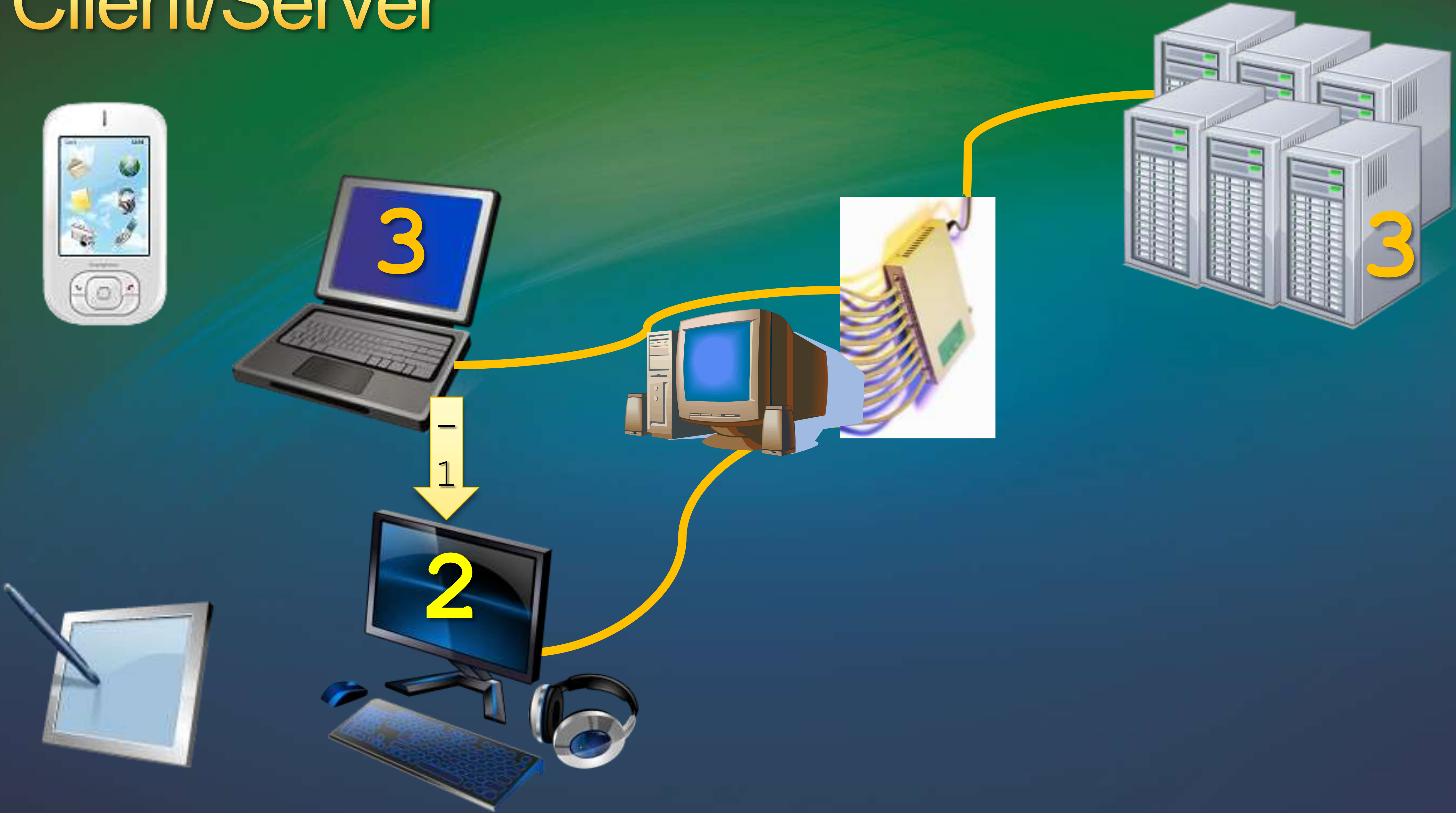
Client/Server



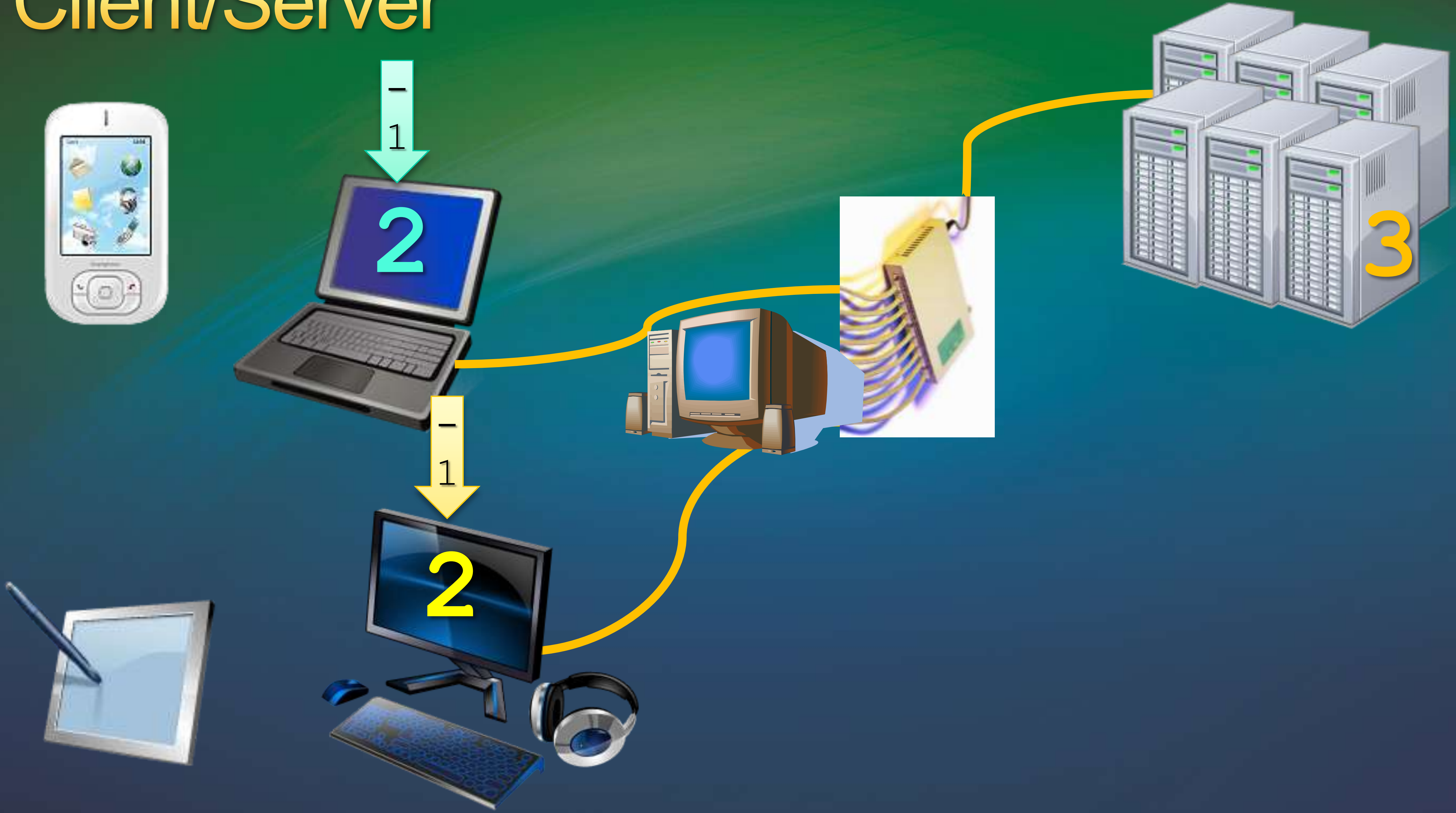
Client/Server



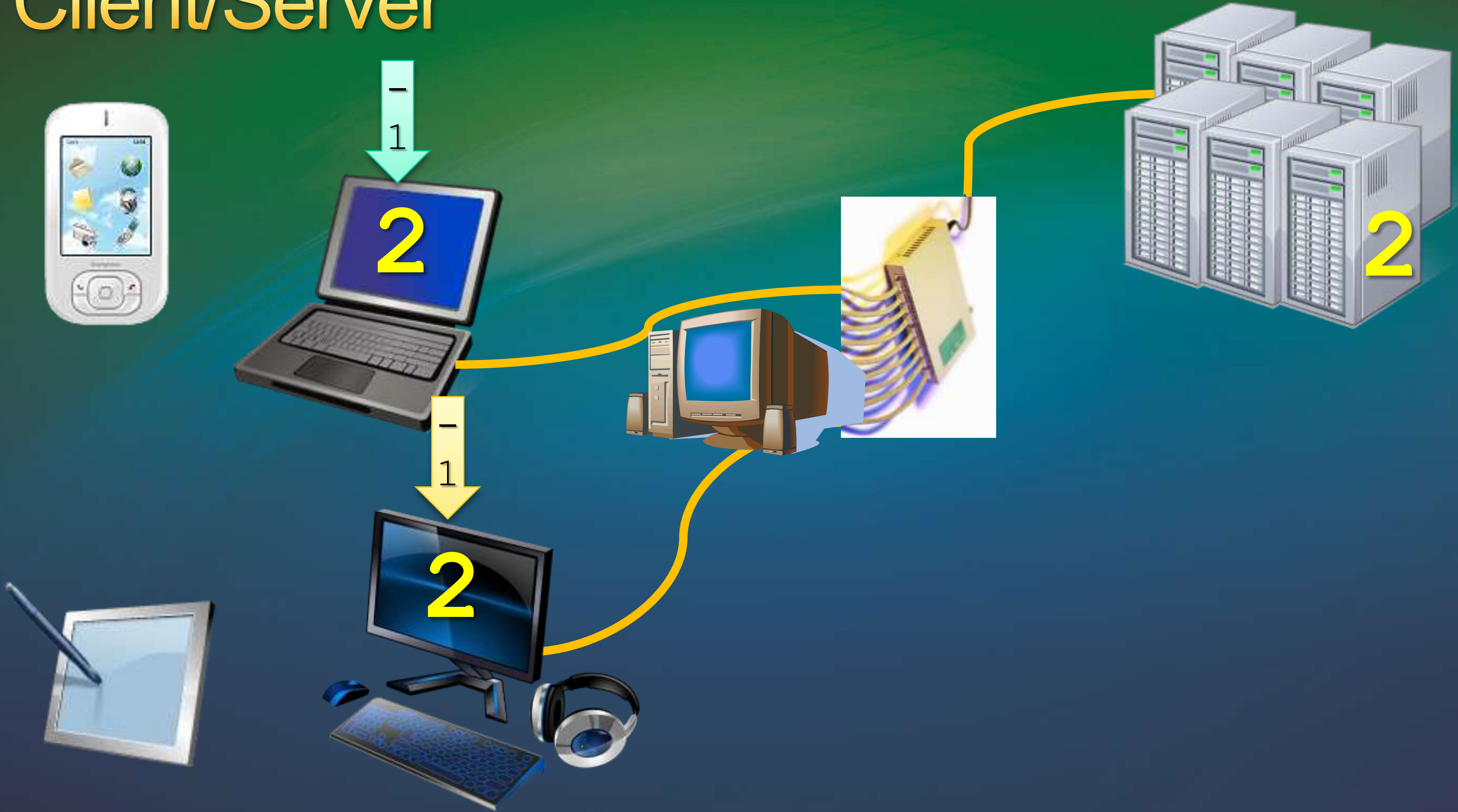
Client/Server



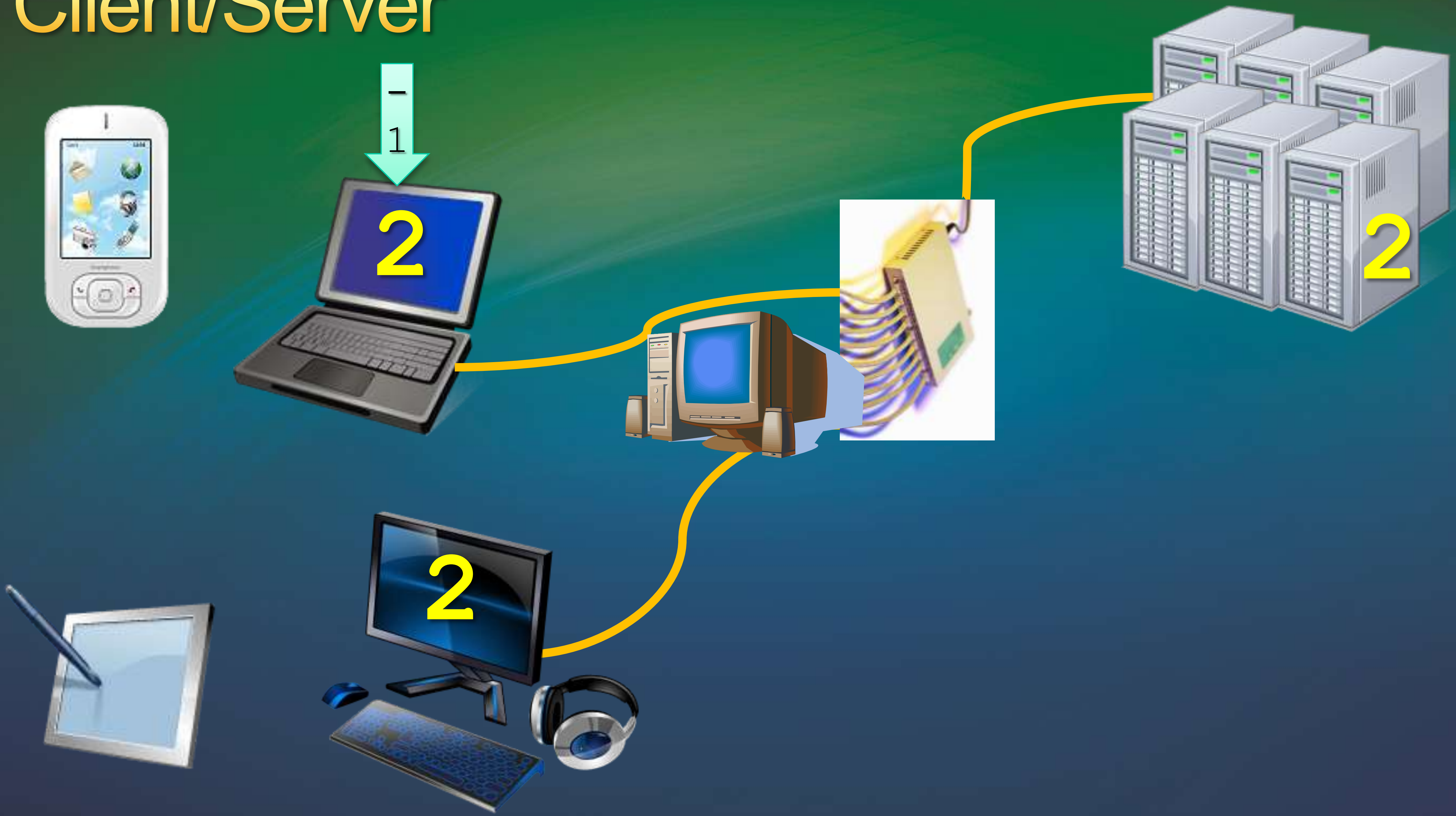
Client/Server



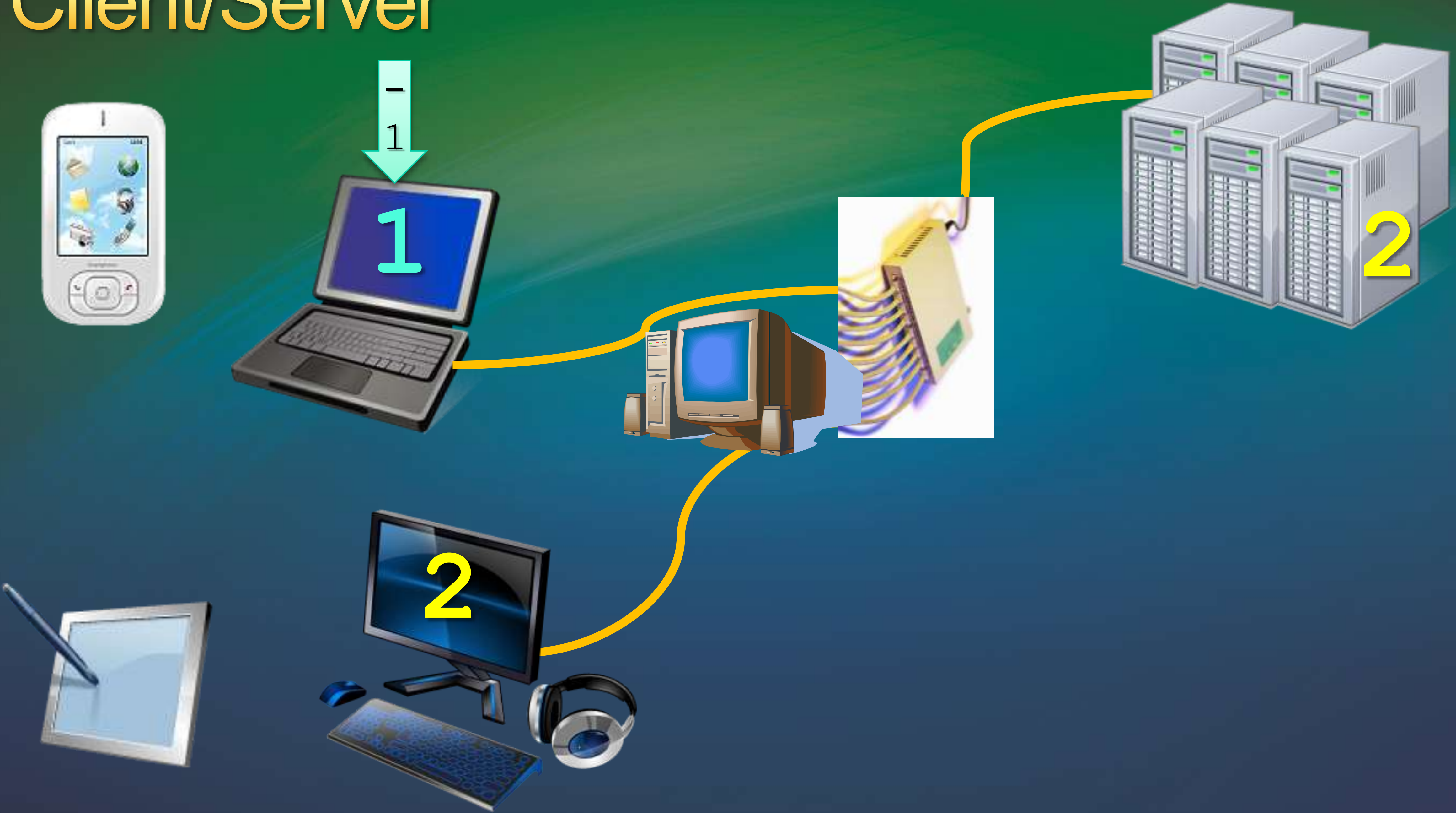
Client/Server



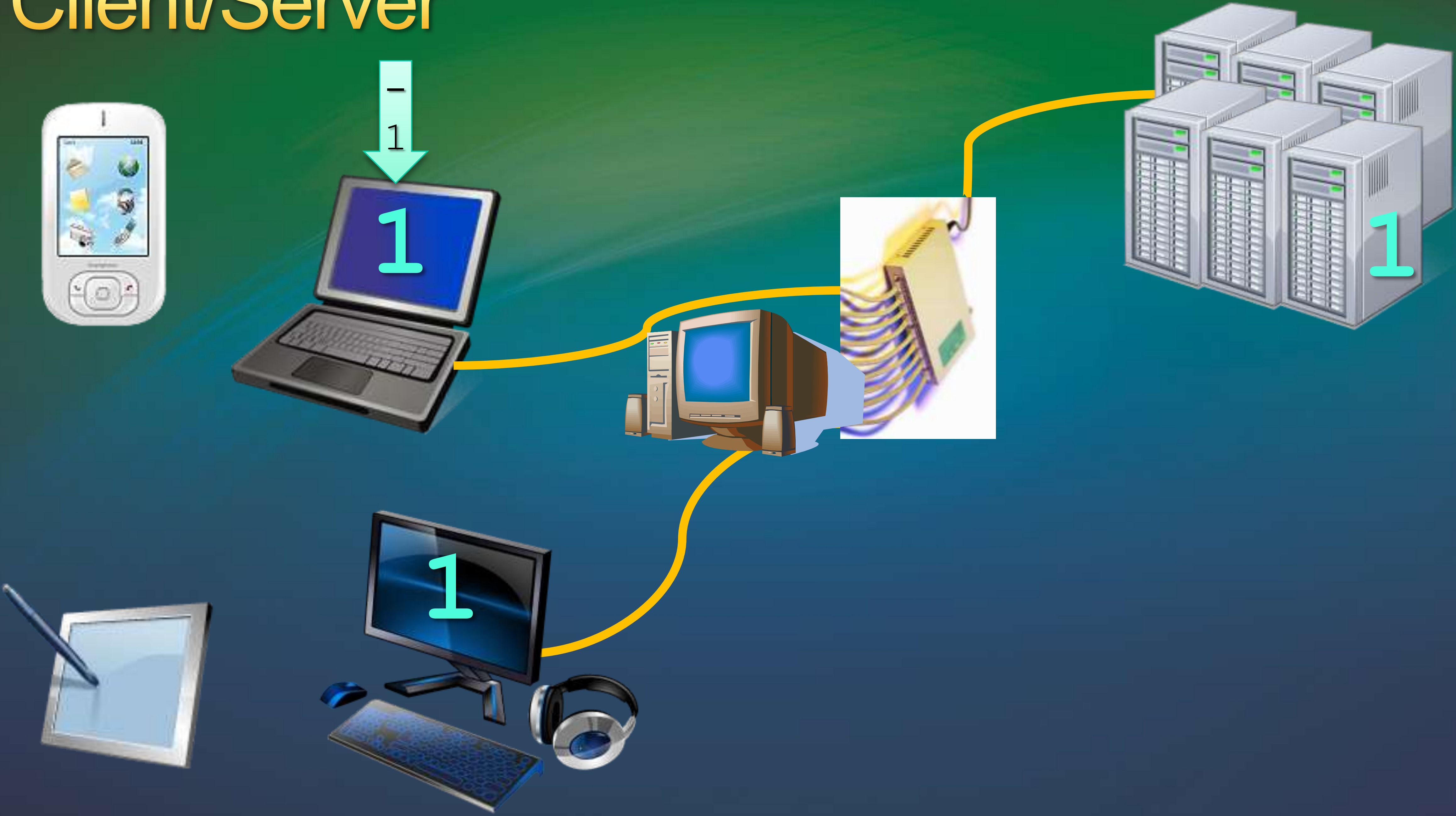
Client/Server



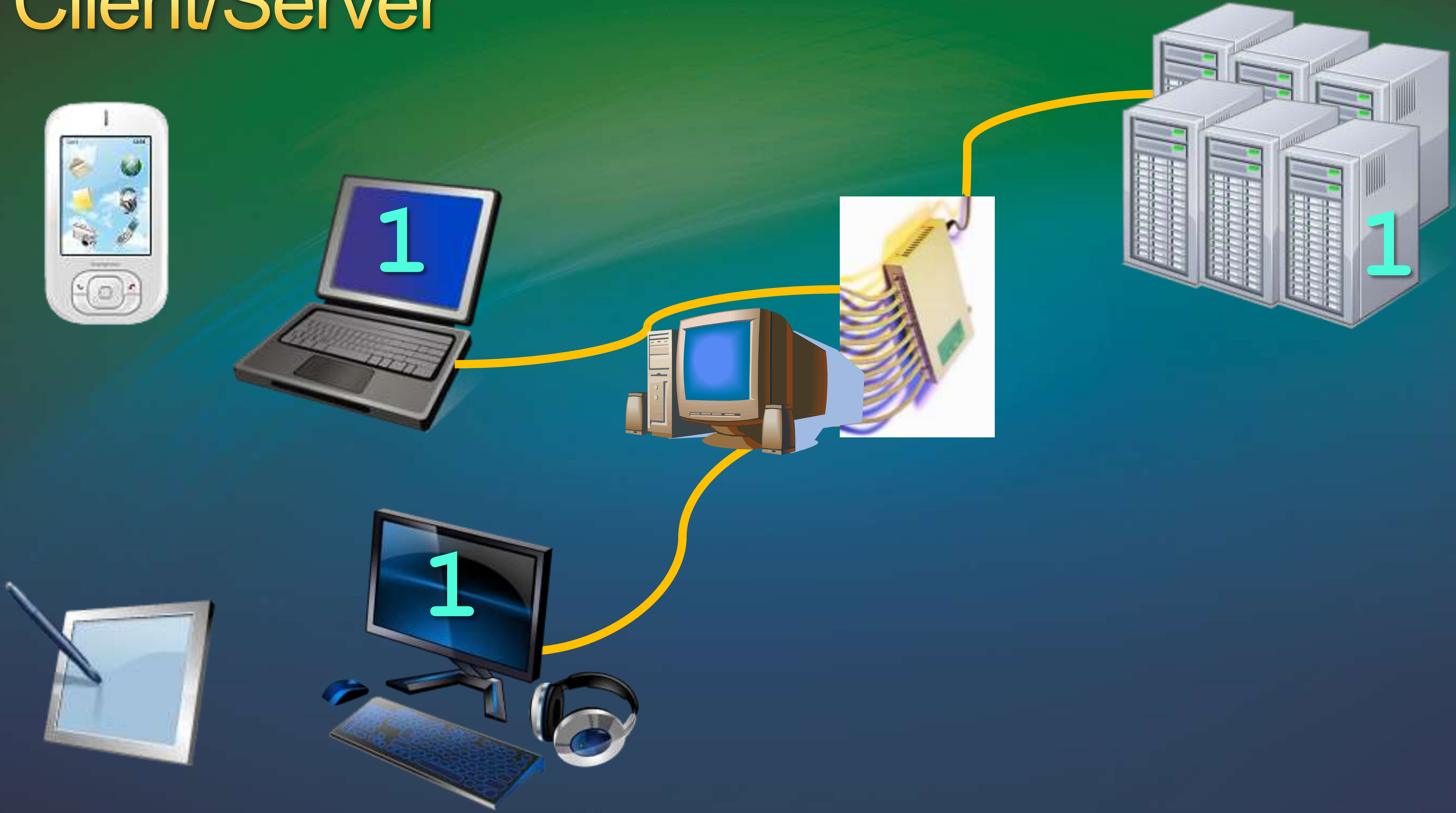
Client/Server



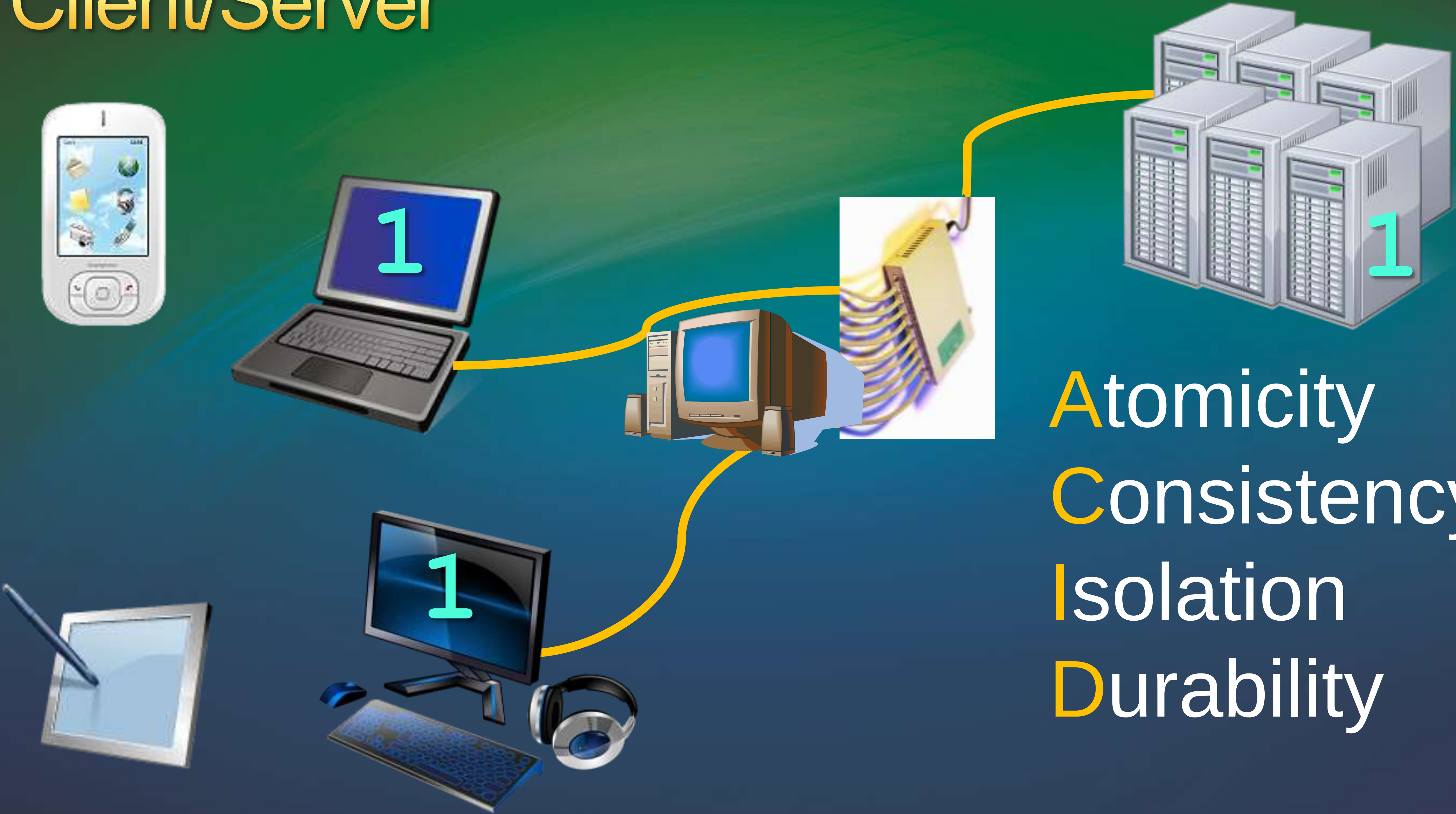
Client/Server



Client/Server



Client/Server



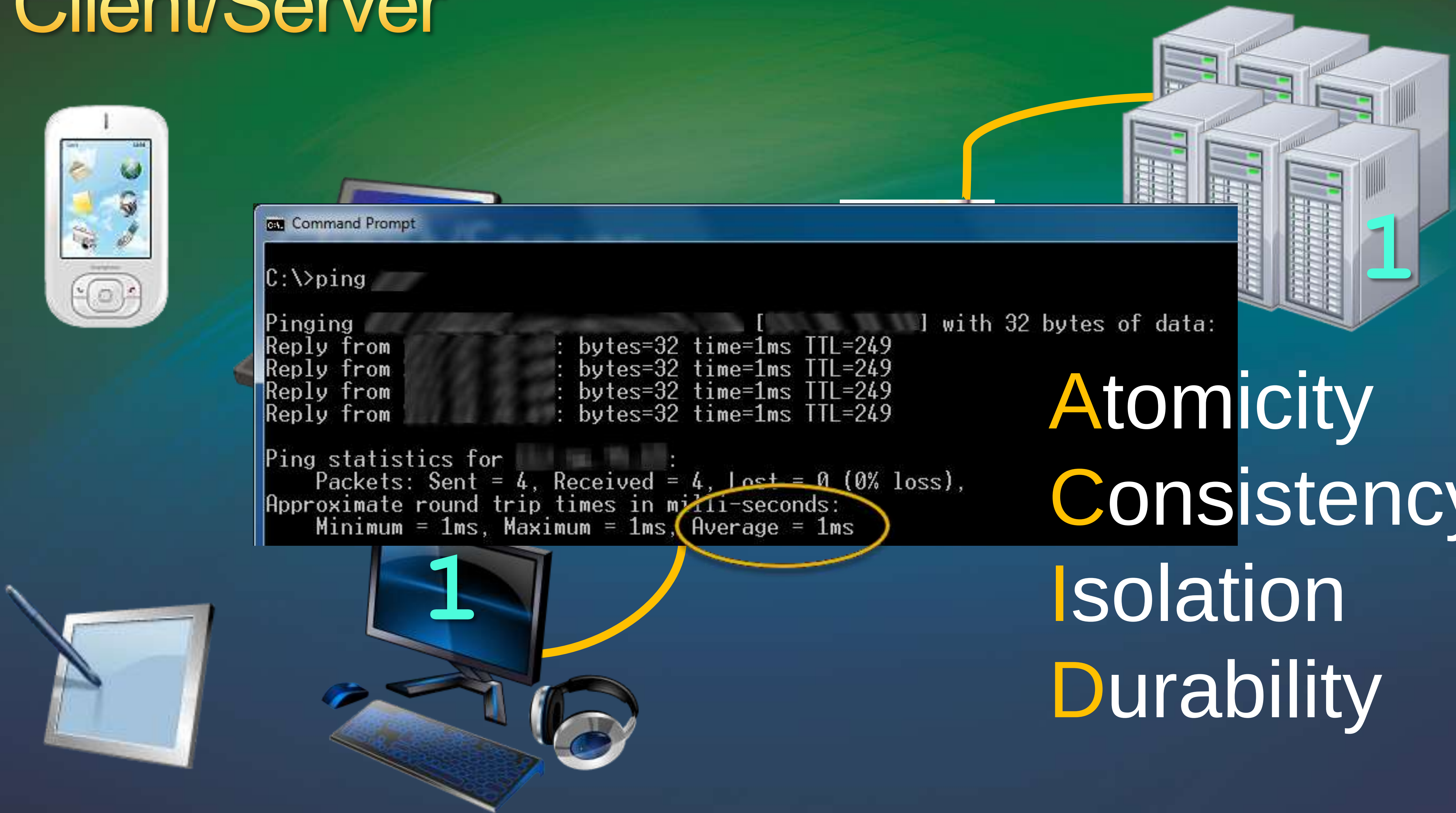
Atomicity

Consistency

Isolation

Durability

Client/Server



```
C:\>ping [redacted]

Pinging [redacted] [[redacted]] with 32 bytes of data:
Reply from [redacted]: bytes=32 time=1ms TTL=249
Reply from [redacted]: bytes=32 time=1ms TTL=249
Reply from [redacted]: bytes=32 time=1ms TTL=249
Reply from [redacted]: bytes=32 time=1ms TTL=249

Ping statistics for [redacted]:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 1ms, Maximum = 1ms, Average = 1ms
```

Atomicity

Consistency

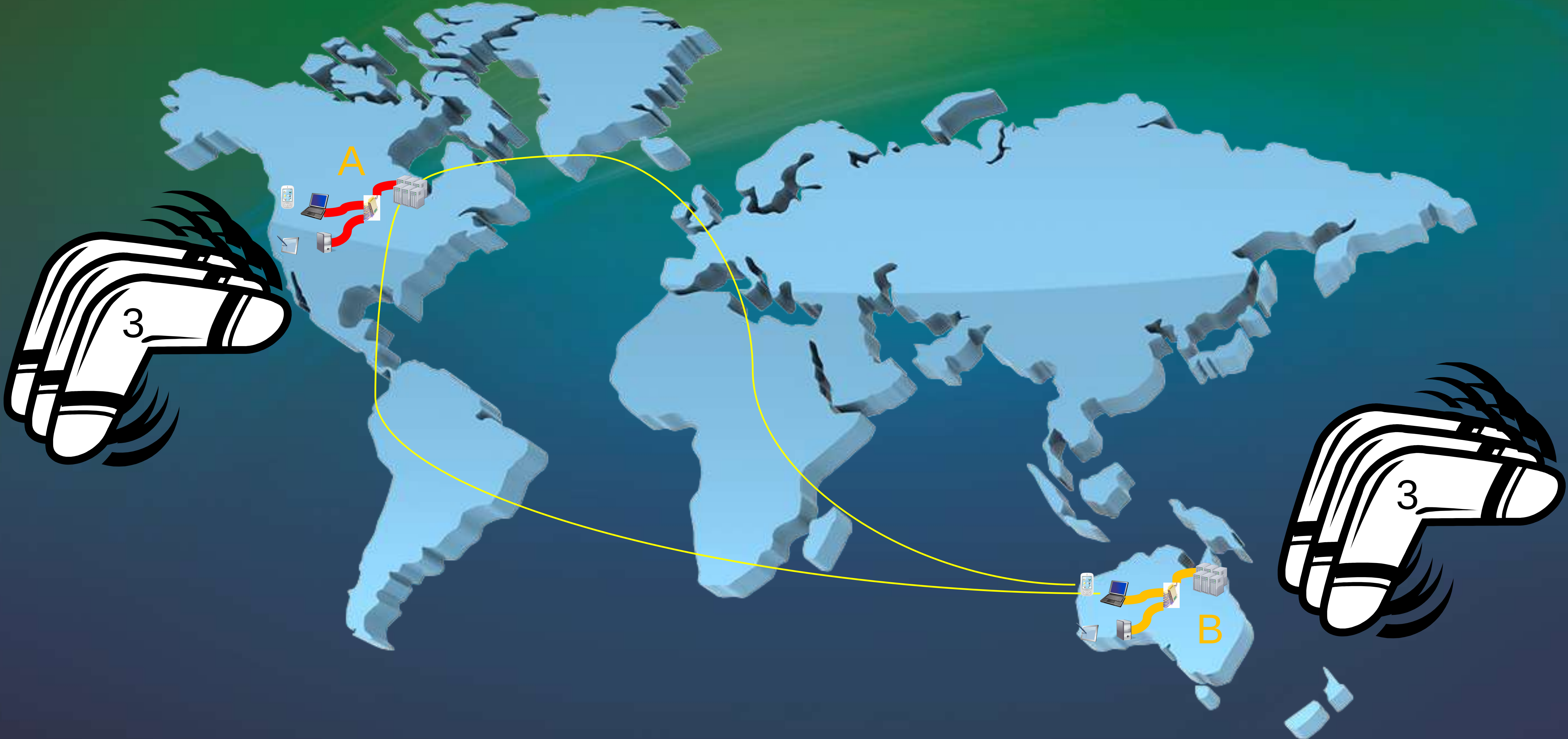
Isolation

Durability

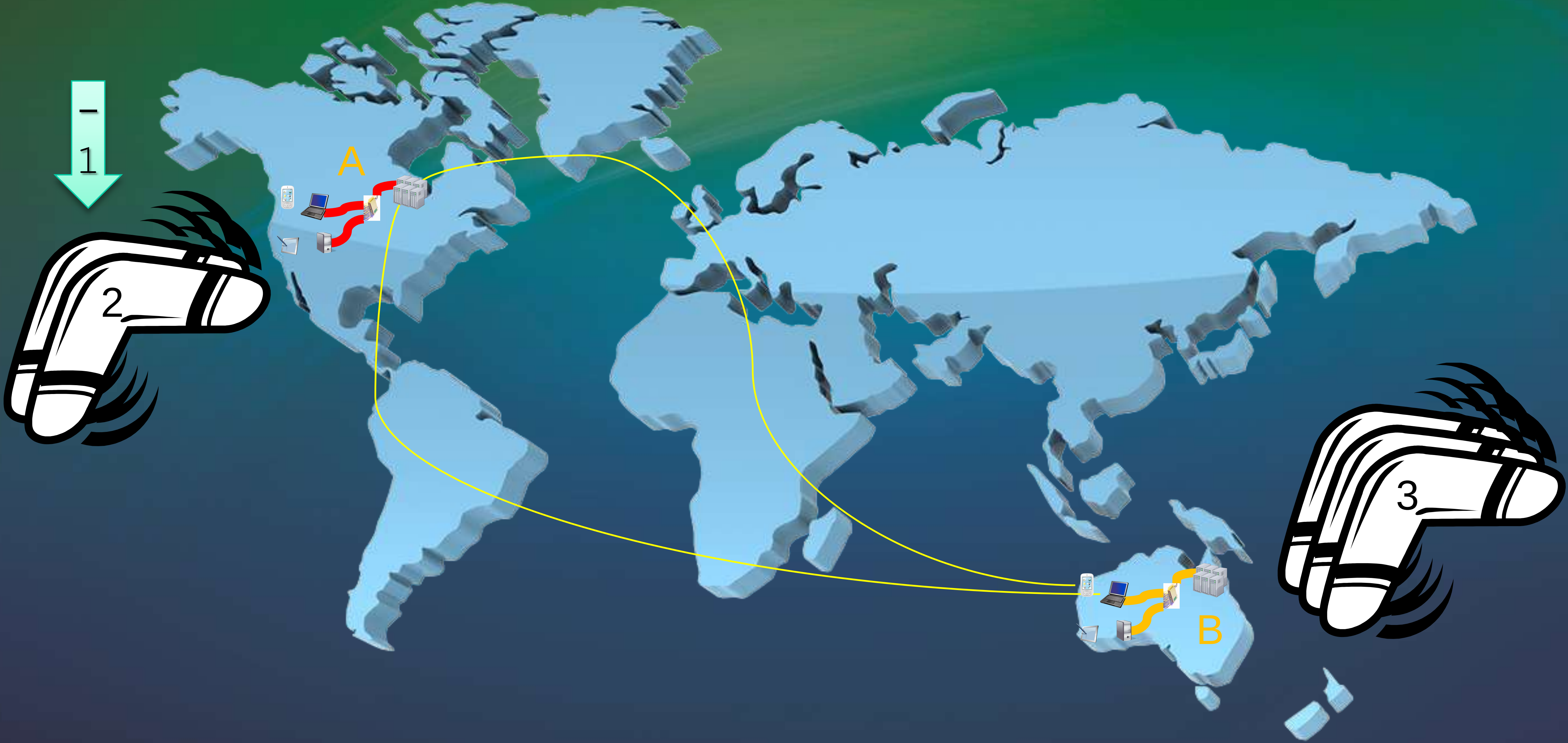
Response time limit

2.7s

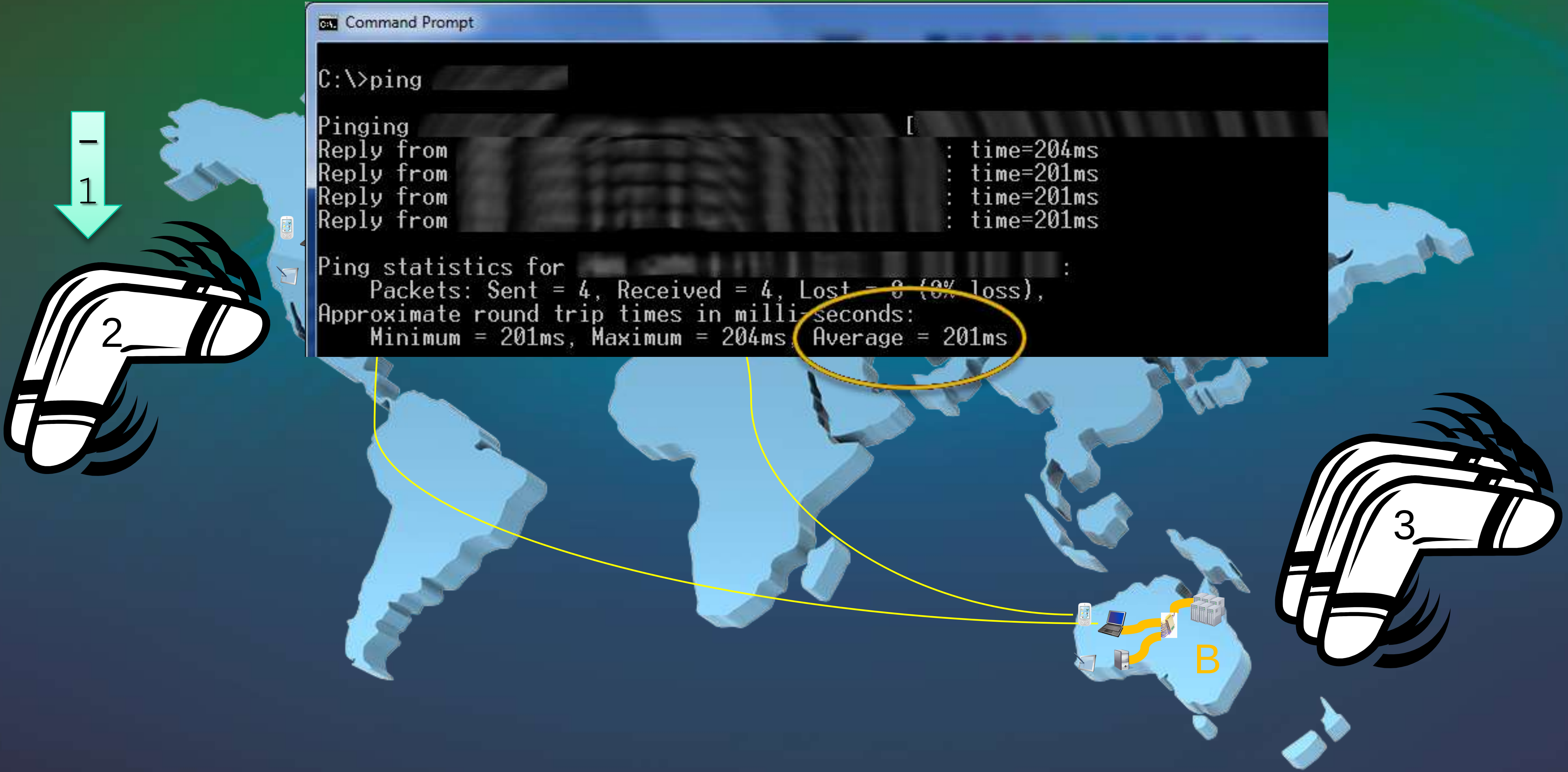
Internet-scale online systems



Internet-scale online systems



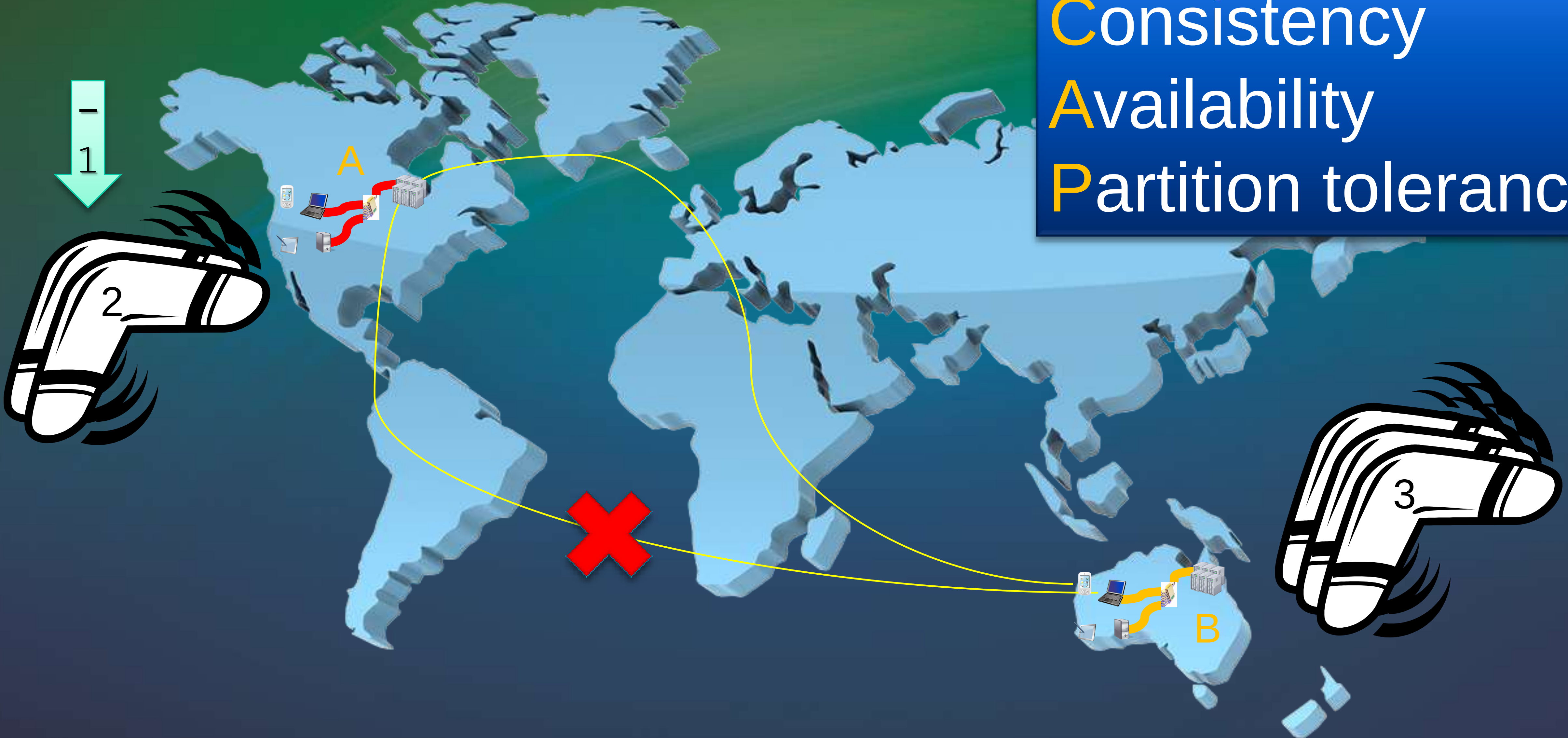
Internet-scale online systems



```
C:\>ping [redacted]

Pinging [redacted] [[redacted]
Reply from [redacted]: time=204ms
Reply from [redacted]: time=201ms
Reply from [redacted]: time=201ms
Reply from [redacted]: time=201ms

Ping statistics for [redacted]:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 201ms, Maximum = 204ms, Average = 201ms
```

Internet-scale online systems

Consistency
Availability
Partition tolerance



Brewer's CAP Theorem

- "You can have at most two of these properties for any shared-data system"

  [Consistency, Availability, tolerance to network Partitions]

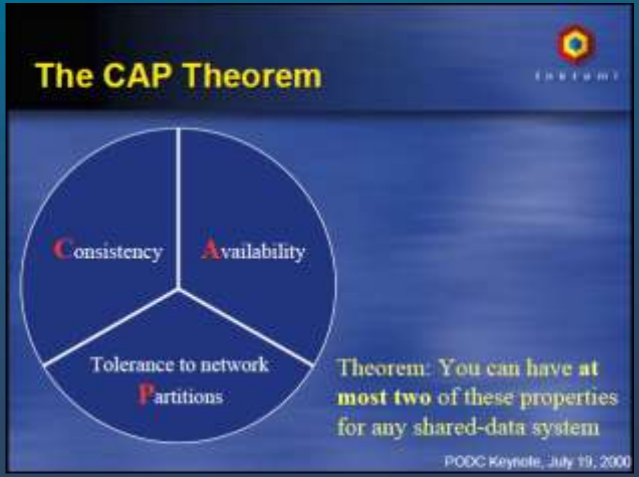
- Towards robust distributed systems
  - Eric Brewer's keynote at Principles of Distributed Computing (PODC) 2000

Brewer's CAP Theorem

- "You can have at most two of these properties for any shared-data system"

[Consistency, Availability, tolerance to network Partitions]

- Towards robust distributed systems
  - Eric Brewer's keynote at Principles of Distributed Computing (PODC) 2000



The CAP Theorem

Consistency

Availability

Tolerance to network Partitions

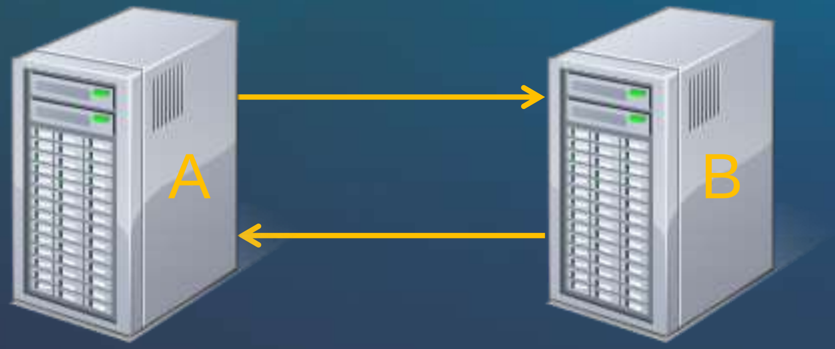
Theorem: You can have at **most two** of these properties for any shared-data system

PODC Keynote, July 19, 2000

Coupling: System A depends on system B

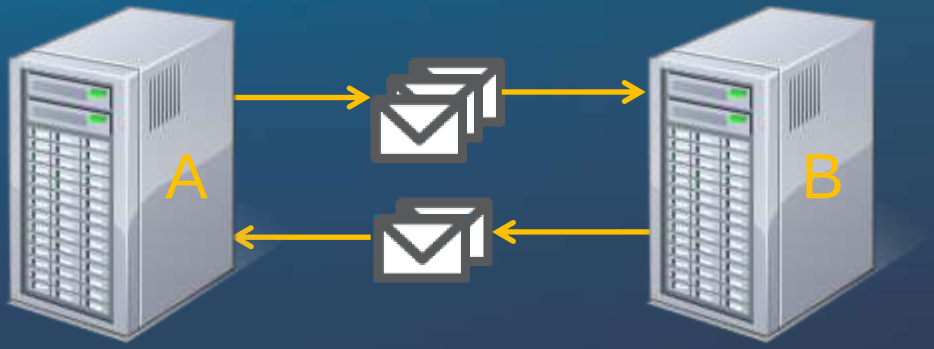
Synchronous

- A is down if B is down
- A is slow if B is slow
- B must grow if A grows



Asynchronous

- A is available independently of B
  - Queue for B may grow
- A has performance independent of B
- A can scale independently of B
  - B must eventually manage the queue

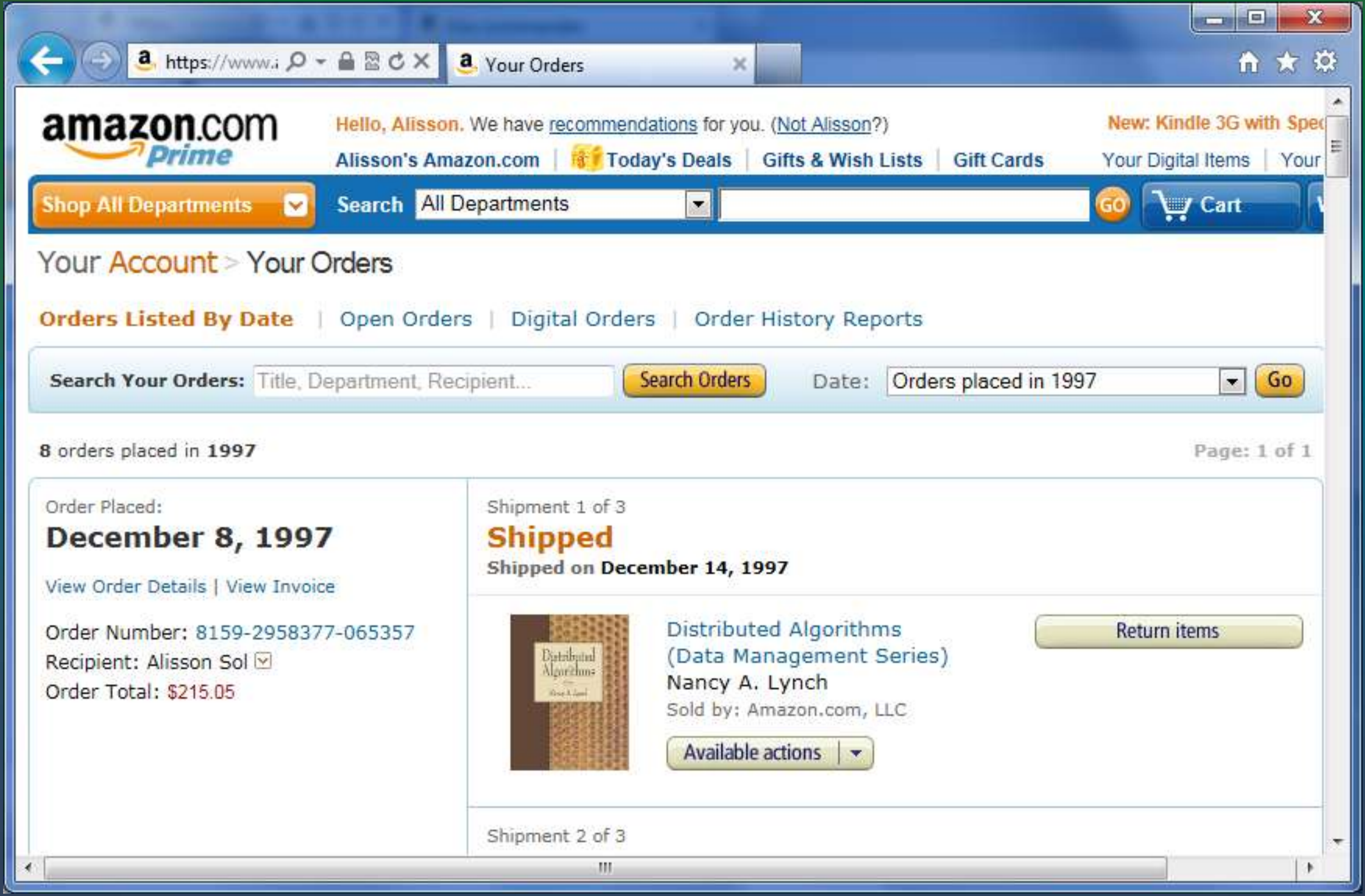


Recommendation

- Conclusion from Brewer's presentation
  - "Winning solution is message-passing clusters"
- Queuing systems
  - Durability, security, delivery, routing, and other functionality
    - MSMQ, WebSphere MQ, Oracle Advanced Queuing, Java Message Service, JBoss Messaging, Kafka, Apache ZooKeeper, Amazon SQS, Azure AppFabric Service Bus, and others
- BASE
  - Basically Available, Soft-state, Eventual consistency

Why this is important for you?

- Architect new systems
  - With local redundancy
  - Using queuing systems when latency is high
- What is local or “atomic” (ACID)?
  - Depends on your problem
    - Create local user account
    - Add item to cart
    - Check out: Charge credit card plus create order
- And some things never need to be consistent...



Hello, Alisson. We have recommendations for you. (Not Alisson?)

Your Account > Your Orders

Orders Listed By Date | Open Orders | Digital Orders | Order History Reports

Search Your Orders: Title, Department, Recipient...

Date: Orders placed in 1997

8 orders placed in **1997**

Page: 1 of 1

Order Placed:
December 8, 1997

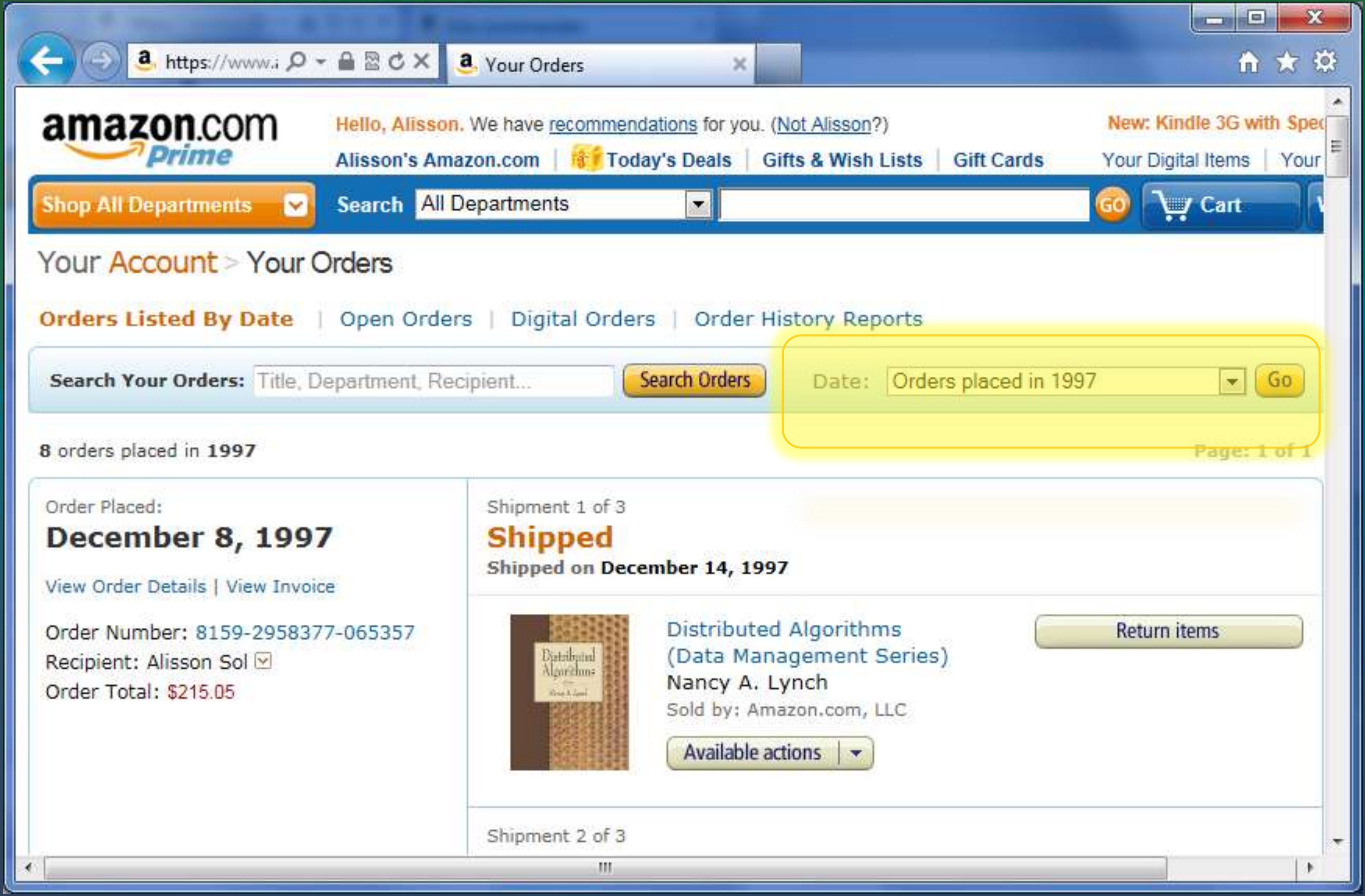
View Order Details | View Invoice

Order Number: 8159-2958377-065357
Recipient: Alisson Sol
Order Total: $215.05

Shipment 1 of 3
Shipped
Shipped on **December 14, 1997**

Distributed Algorithms (Data Management Series)
Nancy A. Lynch
Sold by: Amazon.com, LLC

Shipment 2 of 3



Your Account > Your Orders

Search Your Orders: Title, Department, Recipient...

Date: Orders placed in 1997

8 orders placed in **1997**

Page: 1 of 1

Order Placed:
December 8, 1997

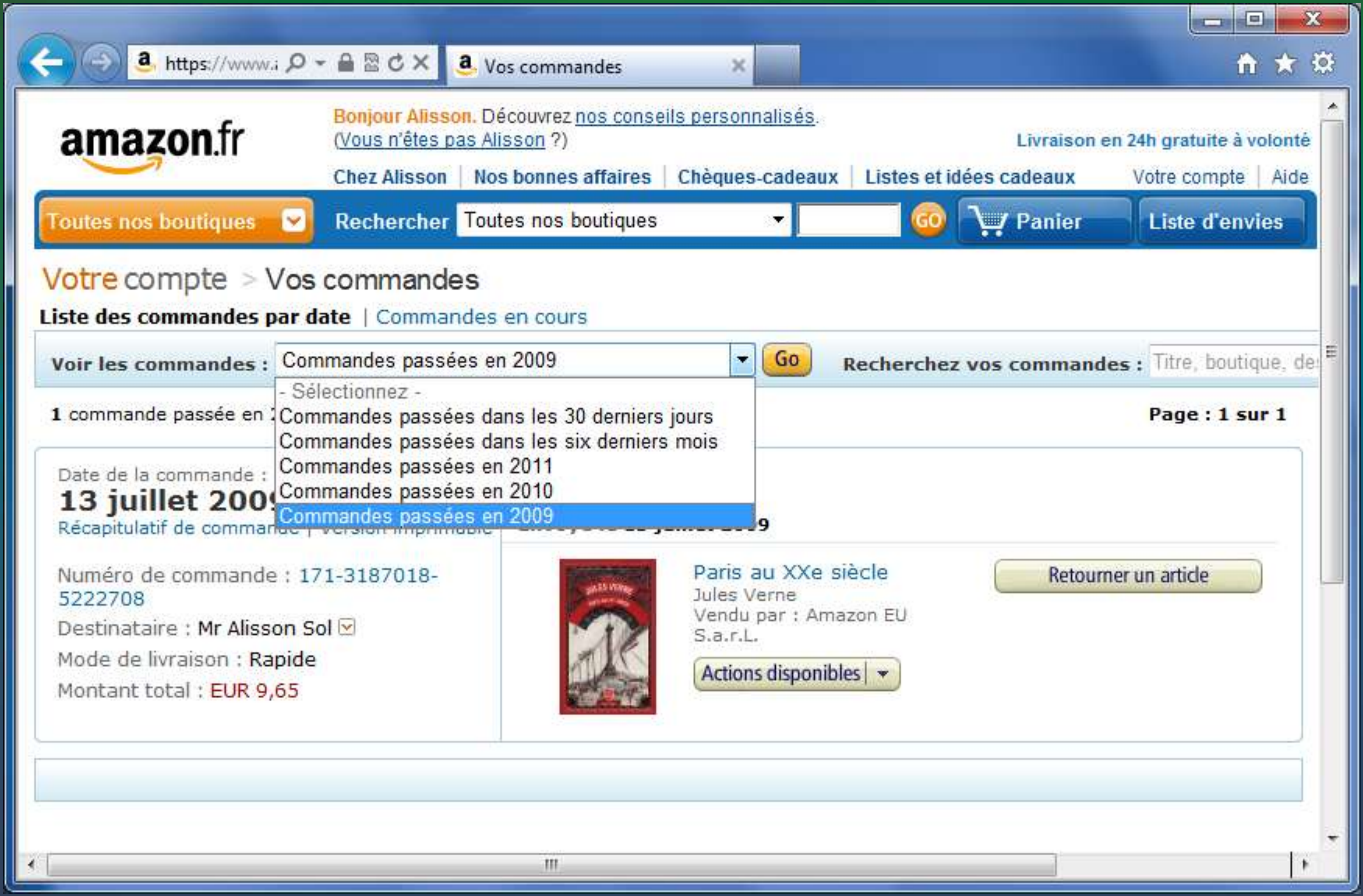
View Order Details | View Invoice

Order Number: 8159-2958377-065357
Recipient: Alisson Sol
Order Total: $215.05

Shipment 1 of 3
Shipped
Shipped on **December 14, 1997**

Distributed Algorithms (Data Management Series)
Nancy A. Lynch
Sold by: Amazon.com, LLC

Shipment 2 of 3



Bonjour Alisson. Découvrez nos conseils personnalisés.
(Vous n'êtes pas Alisson ?)

Livraison en 24h gratuite à volonté

Chez Alisson | Nos bonnes affaires | Chèques-cadeaux | Listes et idées cadeaux | Votre compte | Aide

Rechercher: Toutes nos boutiques

Votre compte > Vos commandes

Liste des commandes par date | Commandes en cours

Voir les commandes : Commandes passées en 2009

- Sélectionnez -
Commandes passées dans les 30 derniers jours
Commandes passées dans les six derniers mois
Commandes passées en 2011
Commandes passées en 2010
Commandes passées en 2009

Recherchez vos commandes : Titre, boutique, de

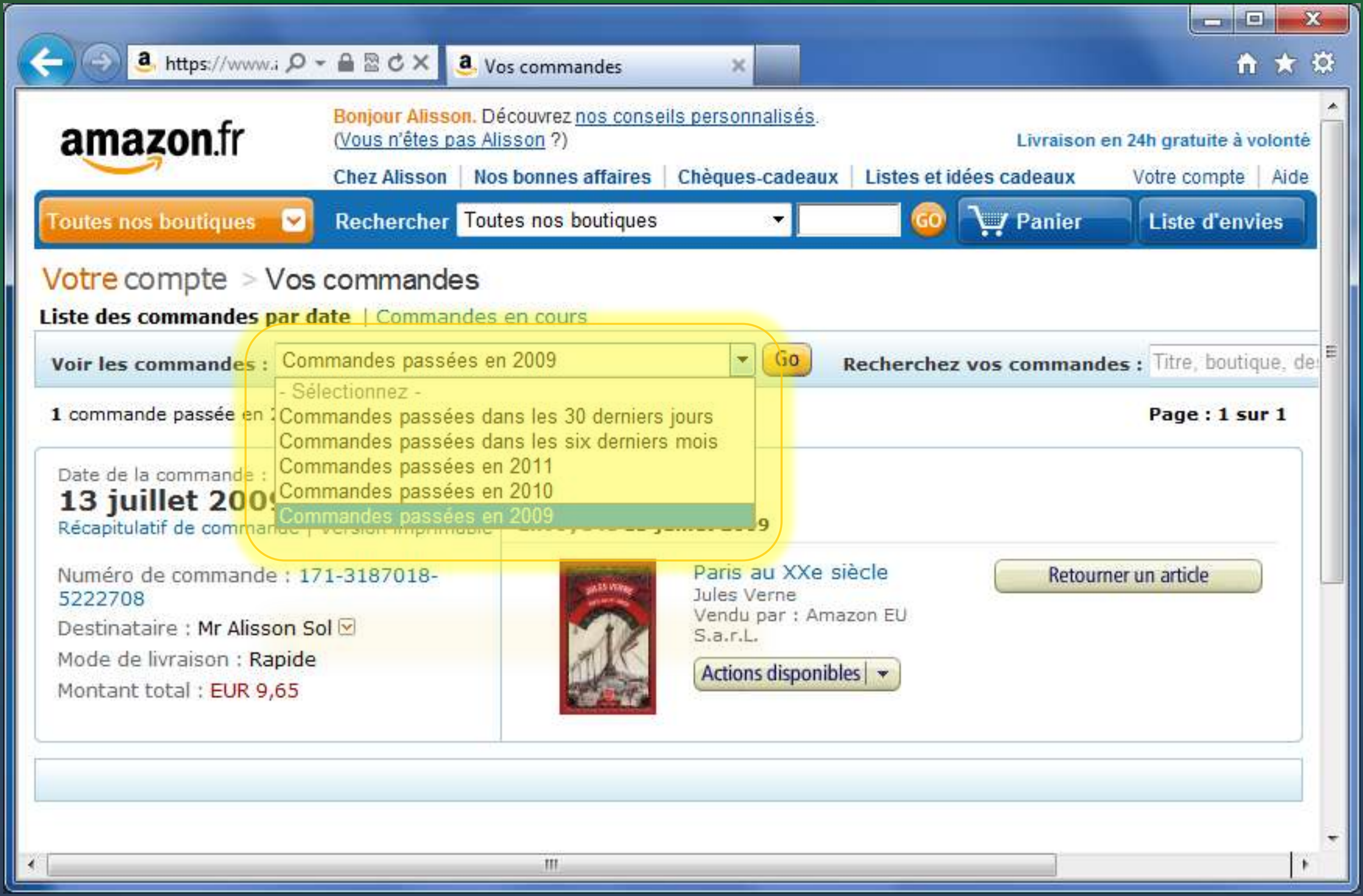
1 commande passée en 2

Page : 1 sur 1

Date de la commande :
13 juillet 200
Récapitulatif de commande | Version imprimable

Numéro de commande : 171-3187018-5222708
Destinataire : Mr Alisson Sol
Mode de livraison : Rapide
Montant total : EUR 9,65

Paris au XXe siècle
Jules Verne
Vendu par : Amazon EU S.a.r.L.



https://www.i

Vos commandes

amazon.fr

Bonjour Alisson. Découvrez nos conseils personnalisés.
(Vous n'êtes pas Alisson ?)

Livraison en 24h gratuite à volonté

Chez Alisson | Nos bonnes affaires | Chèques-cadeaux | Listes et idées cadeaux | Votre compte | Aide

Toutes nos boutiques | Rechercher Toutes nos boutiques | GO | Panier | Liste d'envies

Votre compte > Vos commandes

Liste des commandes par date | Commandes en cours

Voir les commandes : Commandes passées en 2009 | Go | Recherchez vos commandes : Titre, boutique, de

- Sélectionnez -
Commandes passées dans les 30 derniers jours
Commandes passées dans les six derniers mois
Commandes passées en 2011
Commandes passées en 2010
Commandes passées en 2009

1 commande passée en 2

Page : 1 sur 1

Date de la commande :
13 juillet 200
Récapitulatif de commande | Version imprimable

Numéro de commande : 171-3187018-5222708
Destinataire : Mr Alisson Sol
Mode de livraison : Rapide
Montant total : EUR 9,65

Paris au XXe siècle
Jules Verne
Vendu par : Amazon EU S.a.r.L.

Actions disponibles

Retourner un article

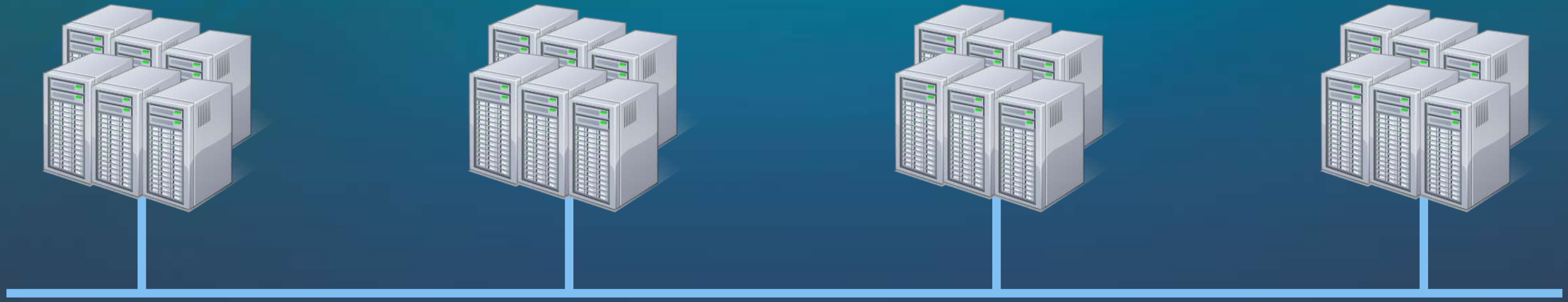
Distributed data: A “key” point

Key	Content
K01	AAOUPA…
K02	ABkJOU…
K03	ACKUij..
…	AD(LJh…

Key	Content
K11	BAOUPA…
K12	DBkJOU…
K13	ECKUij..
…	FD(Lja…

Key	Content
K21	HAOUPA…
K22	MBkJOU…
K23	NCKUij..
…	QD(Lja…

Key	Content
K31	RAOUPA…
K32	SBkJOU…
K33	UCKUij..
…	WD(Lja…



“Content” typically represents several columns of data

Distributed data: A “key” point

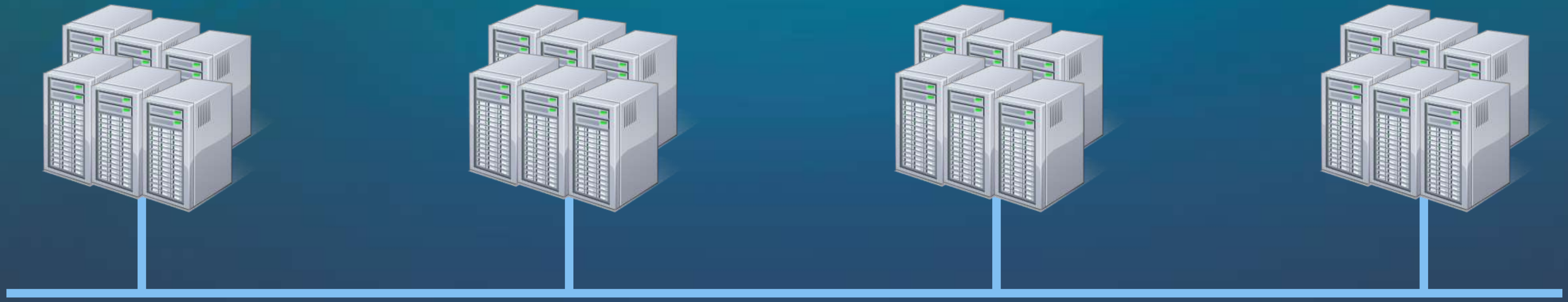
Sharding

Key	Content
K01	AAOUPA…
K02	ABkJOU…
K03	ACKUij..
…	AD(LJh…

Key	Content
K11	BAOUPA…
K12	DBkJOU…
K13	ECKUij..
…	FD(Lja…

Key	Content
K21	HAOUPA…
K22	MBkJOU…
K23	NCKUij..
…	QD(Lja…

Key	Content
K31	RAOUPA…
K32	SBkJOU…
K33	UCKUij..
…	WD(Lja…



“Content” typically represents several columns of data

Processing large datasets

- Which product is selling the most?
  - 10K stores
  - 1K customers/store/day, buying 20 items on average
  - One day
    - 1K customers *10K stores = 10 million receipts
    - 10M receipts * 20 items = 200 million line items
- Equivalent problems
  - Find which page has the most visits in a site, which candidate has the most votes in election, and other equivalent problems

Gray’s laws: Database-centric computing

1. Scientific computing is becoming increasingly data intensive
2. The solution is in a “scale-out” architecture
3. Bring computations to the data, rather than data to the computations
4. Start the design with the “20 queries”
5. Go from “working to working”

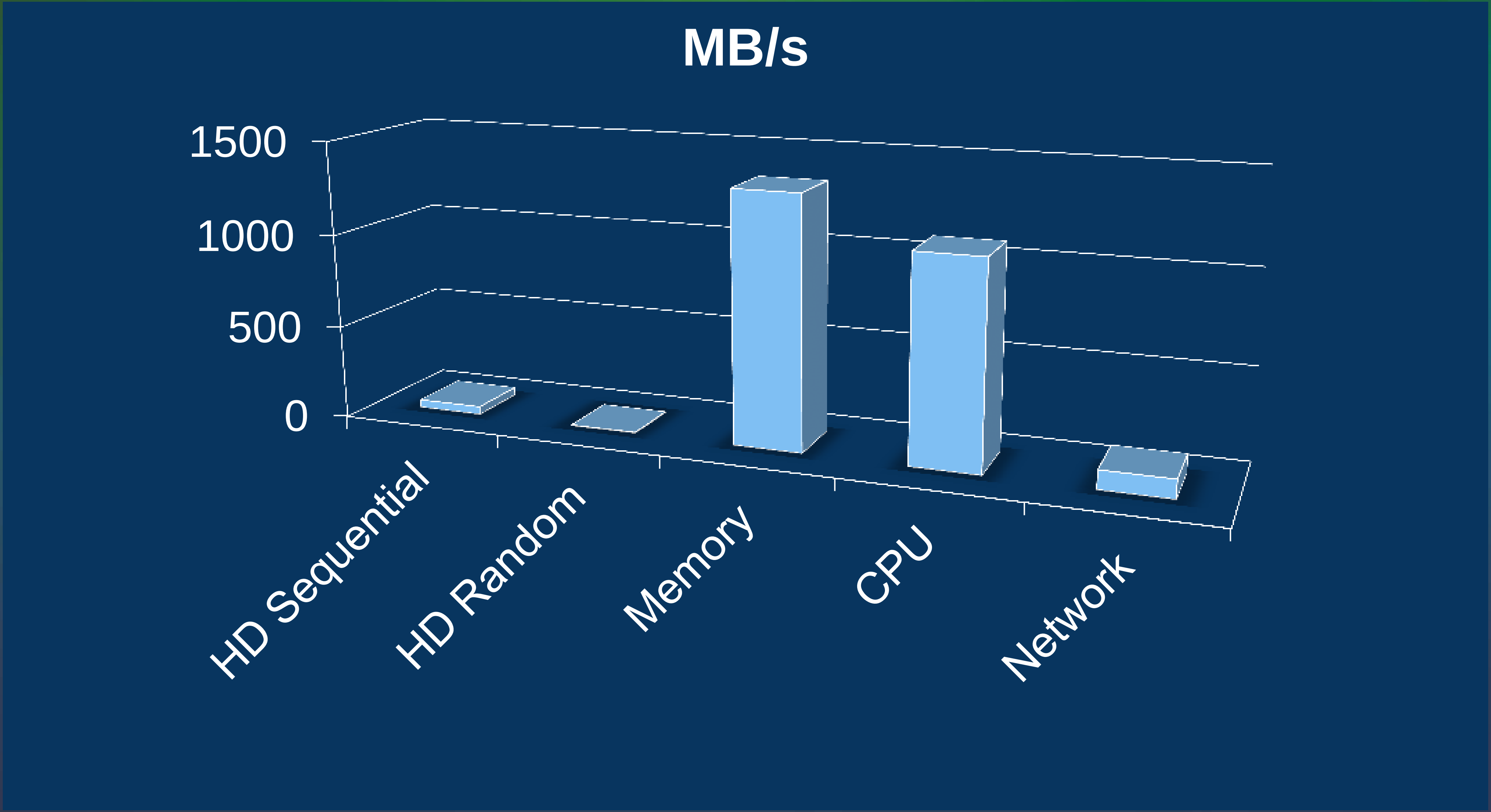
From:
“Gray’s Laws: Database-Centric Computing in Science”
by A.S. Szalay and J.A. Blakeley
In *The Fourth Paradigm–Data-Intensive Scientific Discovery*
edited by Tony Hey, Stewart Tansley, and Krintin Tolle
Microsoft Research, 2009

Reason: The bottleneck

- 1990: HD = 1GB, 4.4MB/s → Read in ~4 minutes
- 2010: HD = 1TB, 100MB/s → Read in ~3 hours

- My laptop (commodity hardware circa 2011)
  - HD Sequential: 42 MB/s
  - HD Random: 3 MB/s
  - Memory transfer: 1,340 MB/s
  - CPU floating point math: 1,093M operations/s
  - Network speed: 1Gbps ~ 100MB/s

Reason: The bottleneck



Distributed processing metamodel

1. Distribute data
2. Distribute processing tasks
3. Combine results

A sample big dataset

- ClueWeb09
  - 1,040,809,705 web pages (~ 10^9 ~= 1G), in 10 languages
  - 5 TB, compressed (25 TB, uncompressed)
    - Uncompressed at 100MB/s → Read in ~3 days
- Web size
  - ~15G pages (from http://www.worldwidewebsize.com/)
    - By extrapolation: Web at 100MB/s → Read in ~45 days

Processing the ClueWeb09 dataset

- Processing time?
  - Considering my laptop and its bottlenecks (individually)

Processing the ClueWeb09 dataset

- Processing time?
  - Considering my laptop and its bottlenecks (individually)

Machines	Processing Time (Minutes)			GB/Machine
	HD.Sequential	HD.Random	Memory	
1	10,477.4	170,002.6	326.0	25,600.0
10	1,047.7	17,000.3	32.6	2,560.0
100	104.8	1,700.0	3.3	256.0
1,000	10.5	170.0	0.3	25.6
10,000	1.0	17.0	0.03	2.6

Processing the ClueWeb09 dataset

- Processing time?
  - Considering my laptop and its bottlenecks (individually)

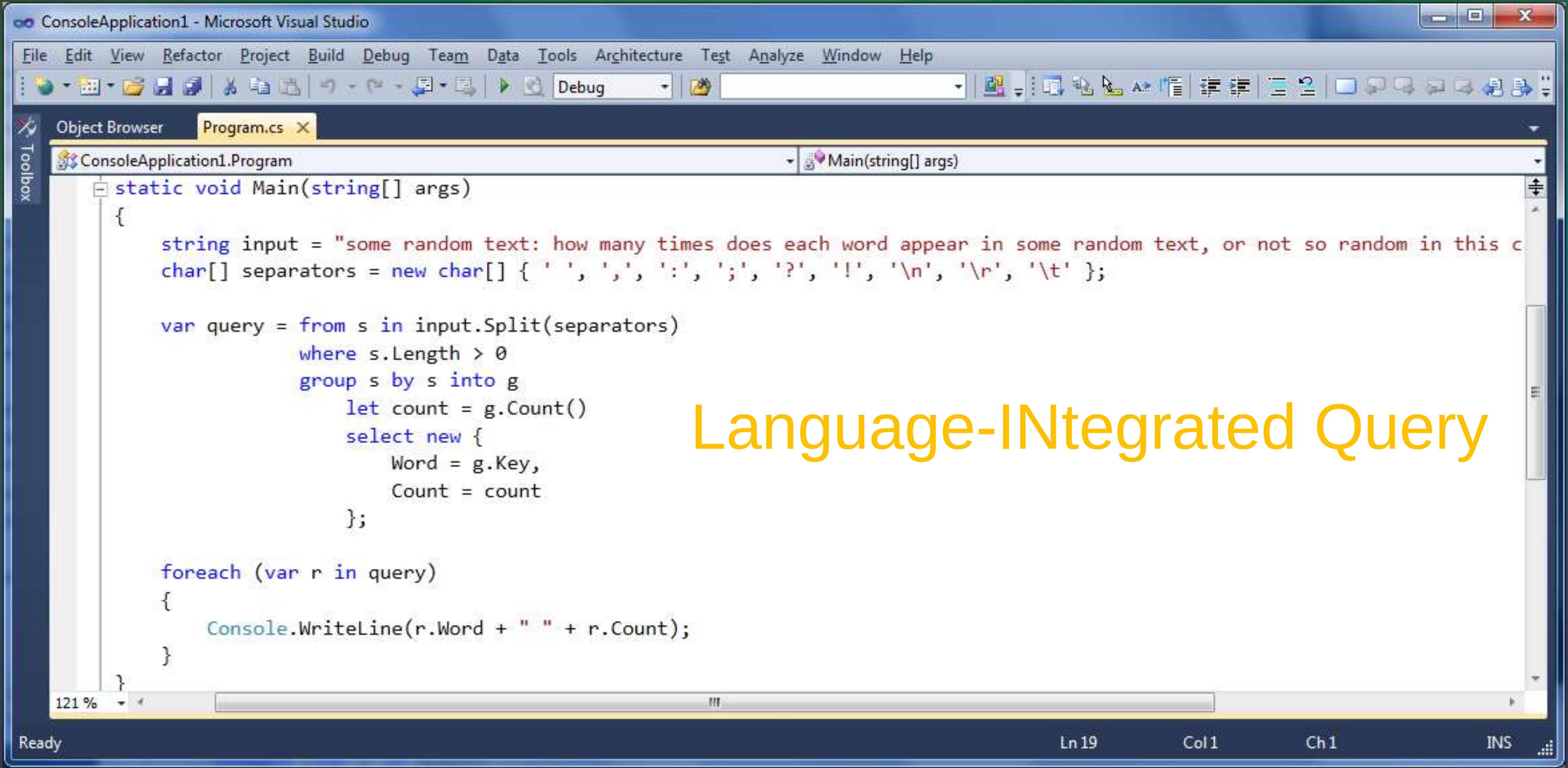
Machines	Processing Time (Minutes)			GB/Machine
	HD.Sequential	HD.Random	Memory	
1	10,477.4	170,002.6	326.0	25,600.0
10	1,047.7	17,000.3	32.6	2,560.0
100	104.8	1,700.0	3.3	256.0
1,000	10.5	170.0	0.3	25.6
10,000	1.0	17.0	0.03	2.6

Processing the ClueWeb09 dataset

- Processing time?
  - Considering my laptop and its bottlenecks (individually)

Machines	Processing Time (Minutes)			GB/Machine
	HD.Sequential	HD.Random	Memory	
1	10,477.4	170,002.6	326.0	25,600.0
10	1,047.7	17,000.3	32.6	2,560.0
100	104.8	1,700.0	3.3	256.0
1,000	10.5	170.0	0.3	25.6
10,000	1.0	17.0	0.03	2.6

How to count words from an input string?



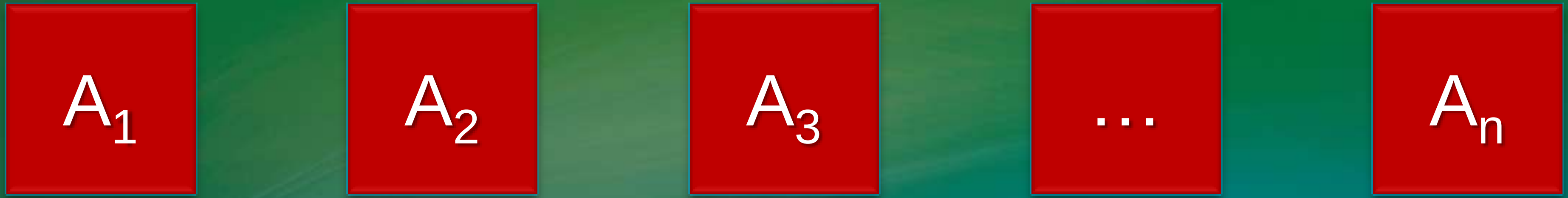
```
static void Main(string[] args)
{
    string input = "some random text: how many times does each word appear in some random text, or not so random in this c
    char[] separators = new char[] { ' ', ',', ':', ';', '?', '!', '\n', '\r', '\t' };

    var query = from s in input.Split(separators)
                where s.Length > 0
                group s by s into g
                    let count = g.Count()
                    select new {
                        Word = g.Key,
                        Count = count
                    };

    foreach (var r in query)
    {
        Console.WriteLine(r.Word + " " + r.Count);
    }
}
```

Language-INtegrated Query

Another option

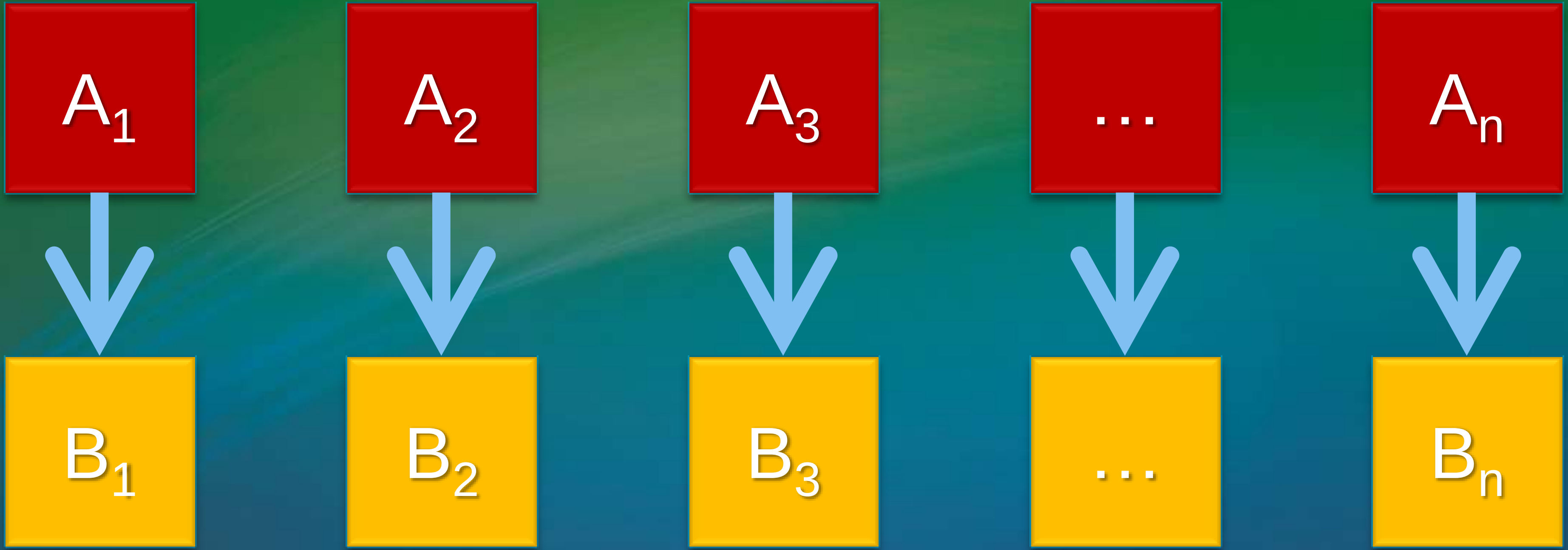
- Functional programming concepts: Map, fold
- Easy to distribute

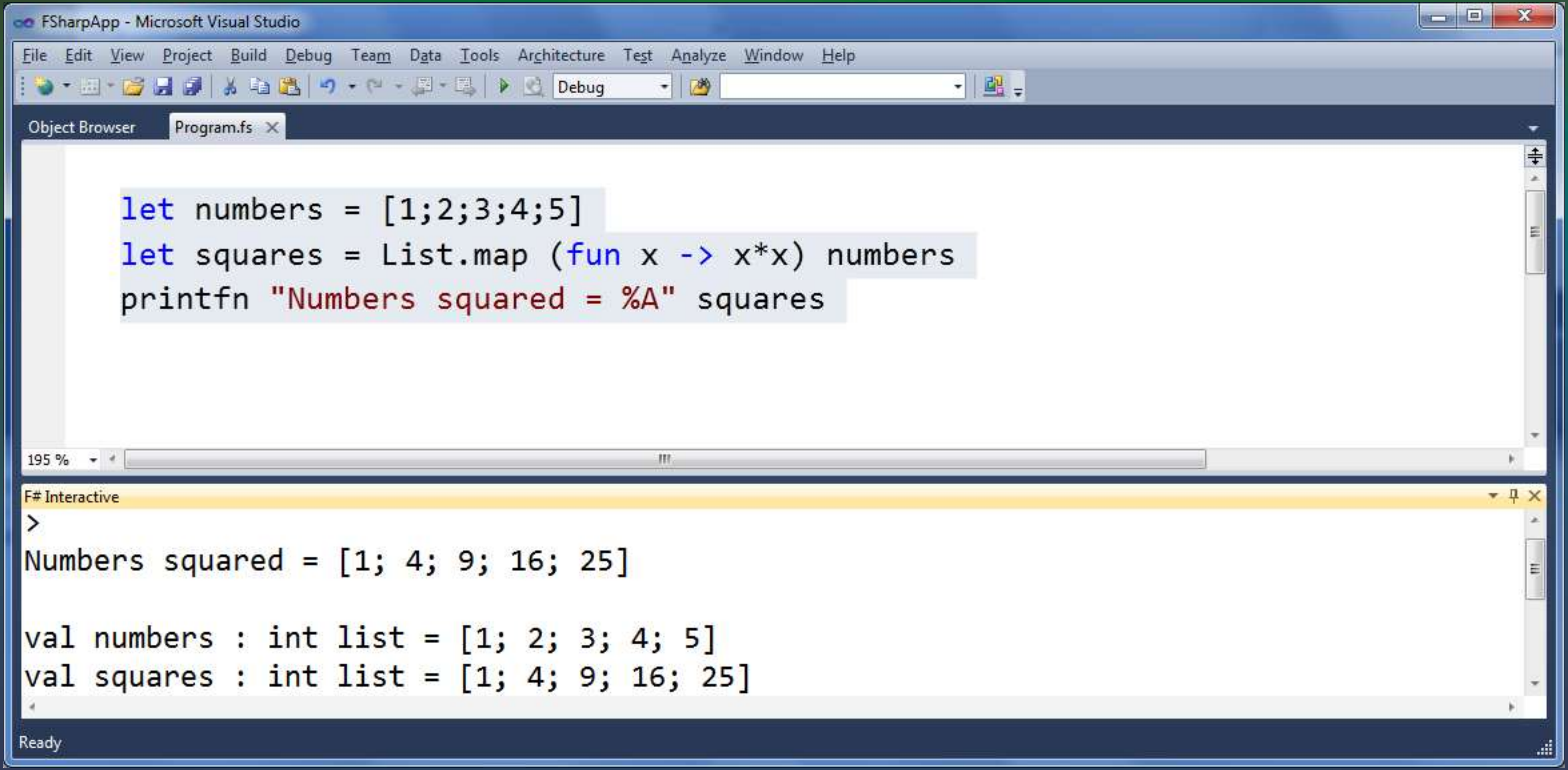
Map



A_1 A_2 A_3 … A_n

Map

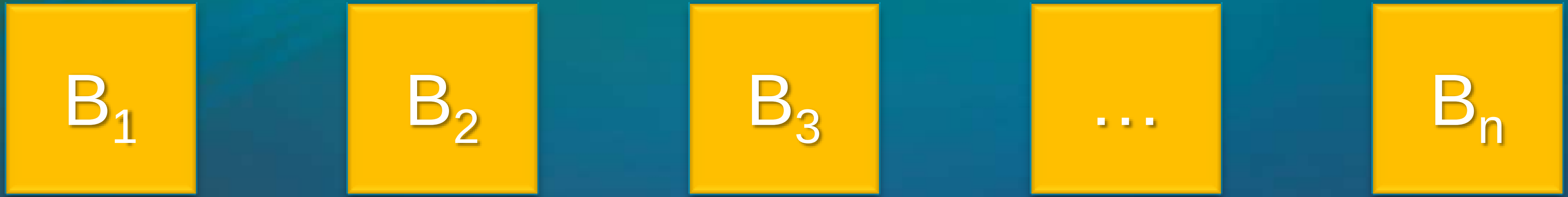



FSharpApp - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Architecture Test Analyze Window Help
Debug
Object Browser
Program.fs
195 %
F# Interactive
Ready

```
let numbers = [1;2;3;4;5]
let squares = List.map (fun x -> x*x) numbers
printfn "Numbers squared = %A" squares
```

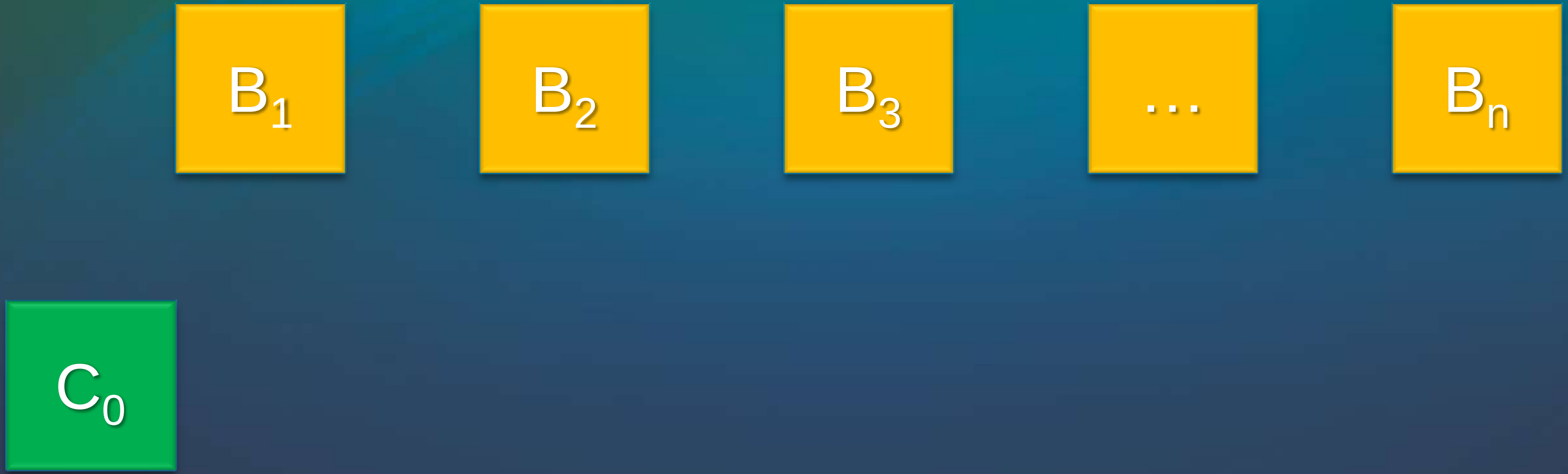
```
>
Numbers squared = [1; 4; 9; 16; 25]

val numbers : int list = [1; 2; 3; 4; 5]
val squares : int list = [1; 4; 9; 16; 25]
```

Fold (reduce)

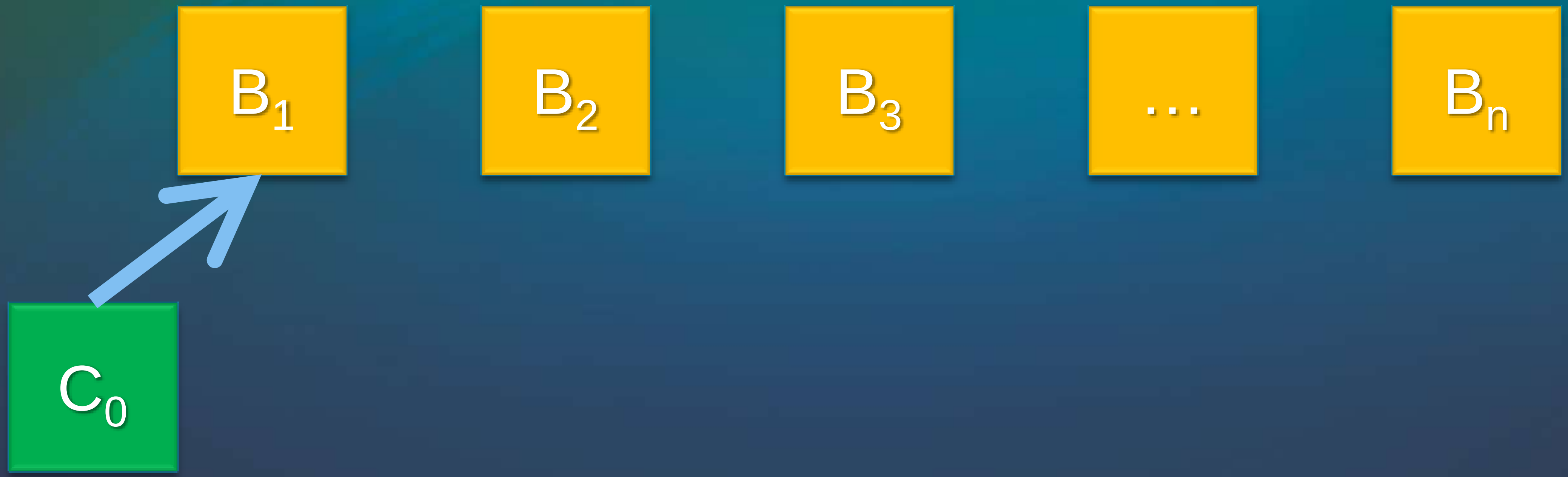


B_1 B_2 B_3 … B_n

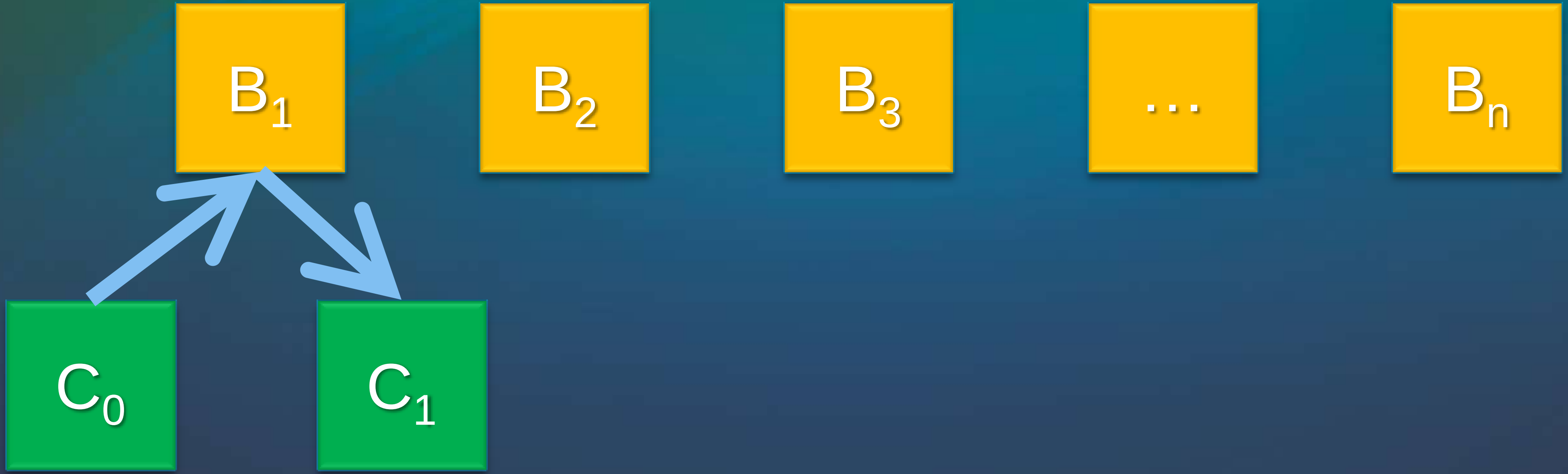
Fold (reduce)



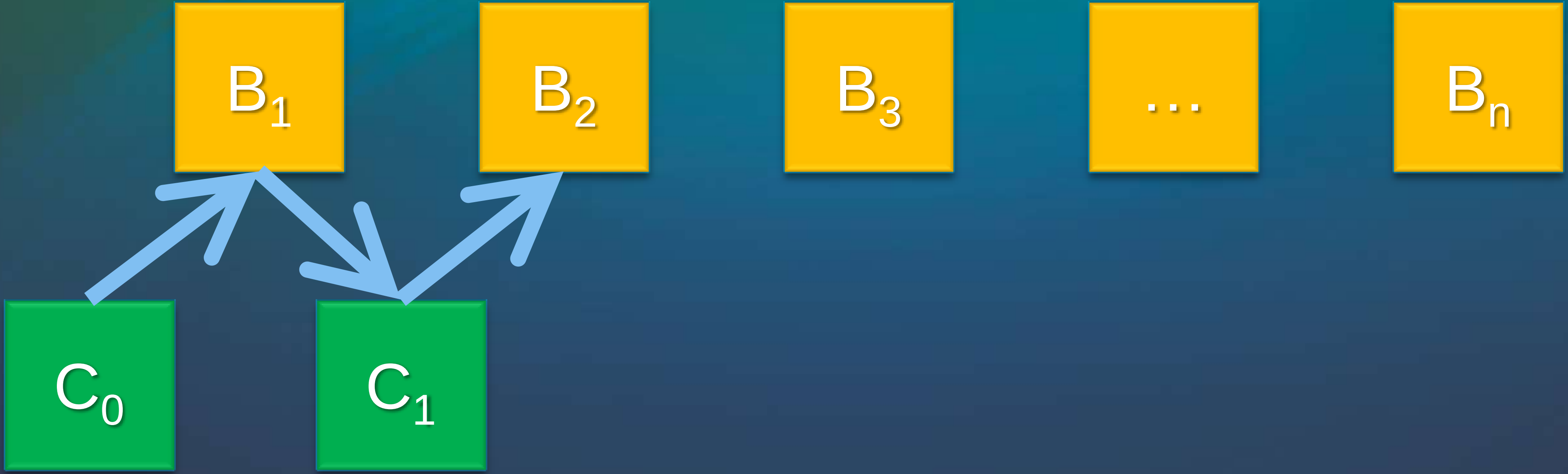
Fold (reduce)



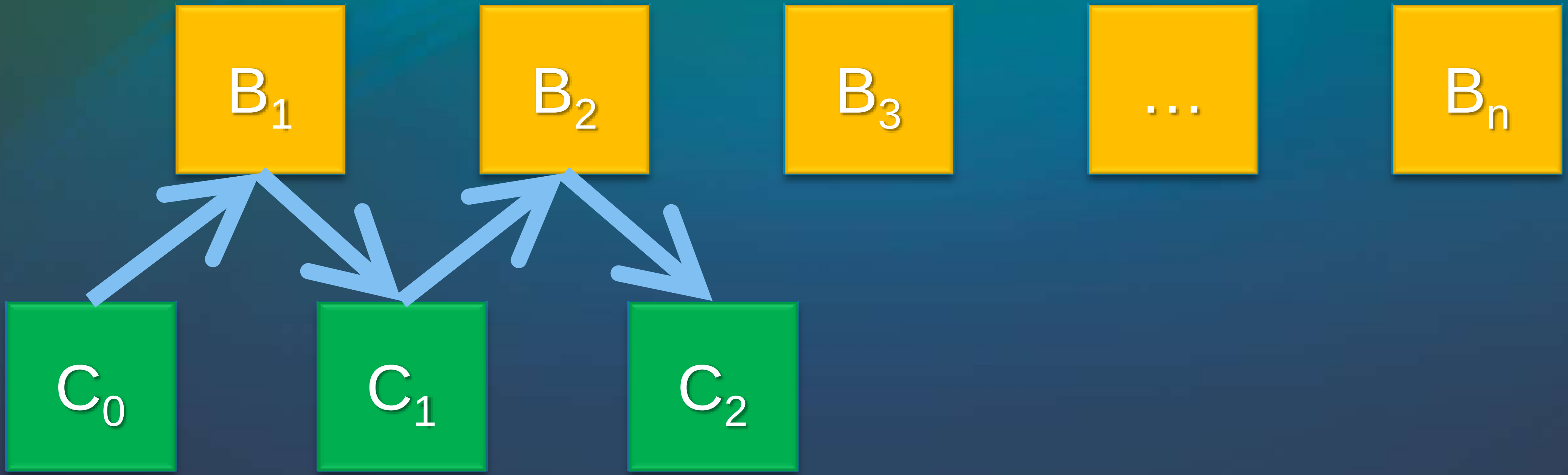
Fold (reduce)



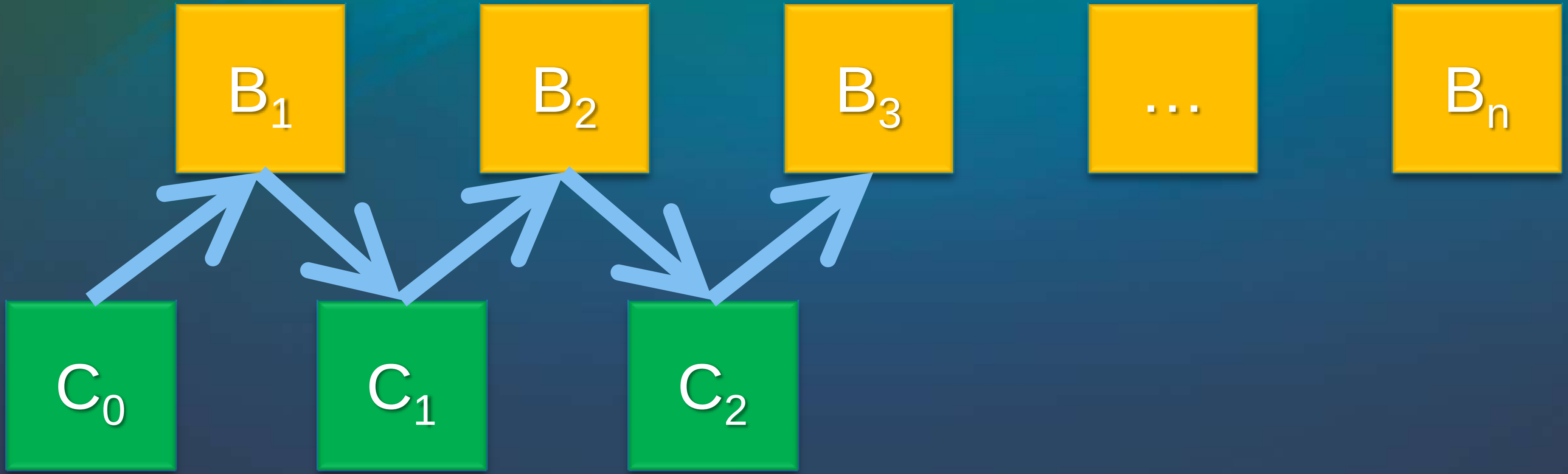
Fold (reduce)



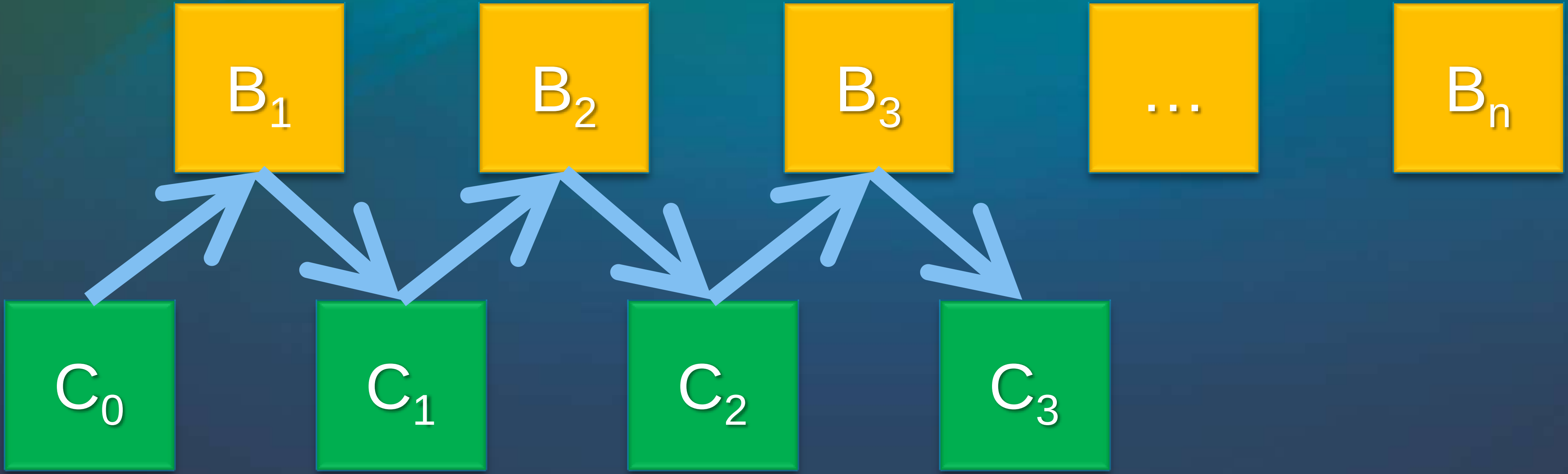
Fold (reduce)



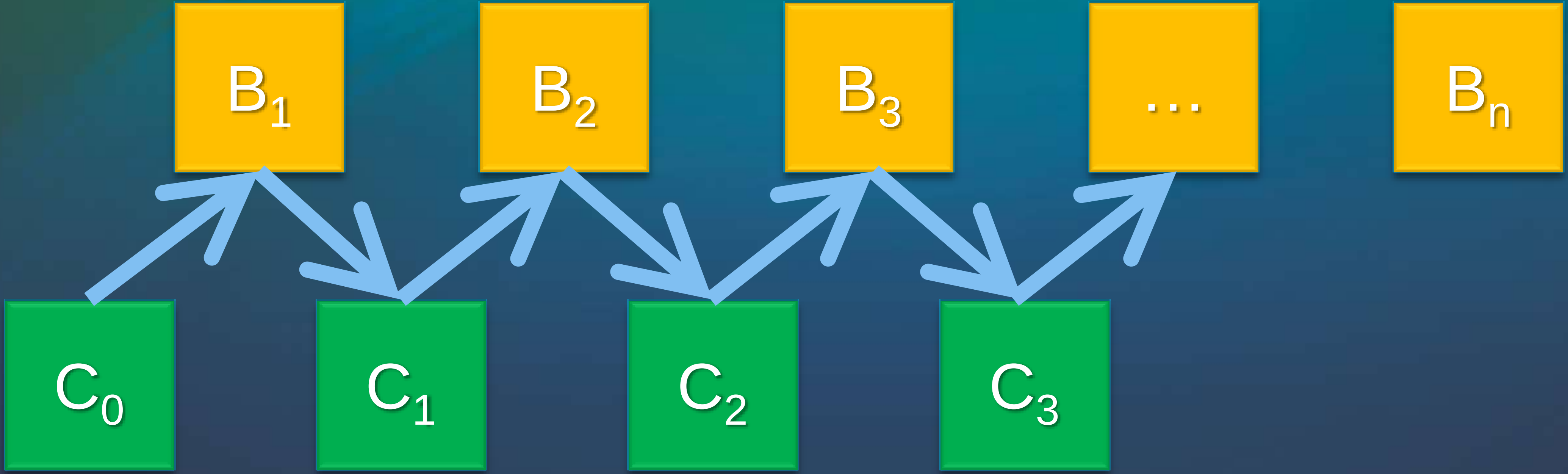
Fold (reduce)



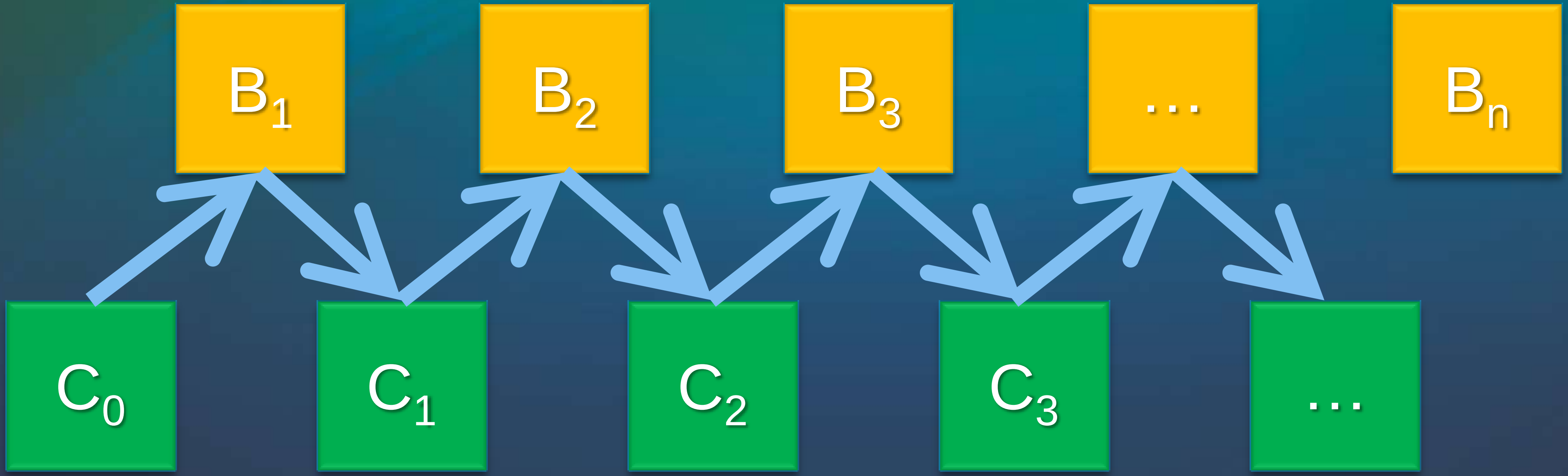
Fold (reduce)



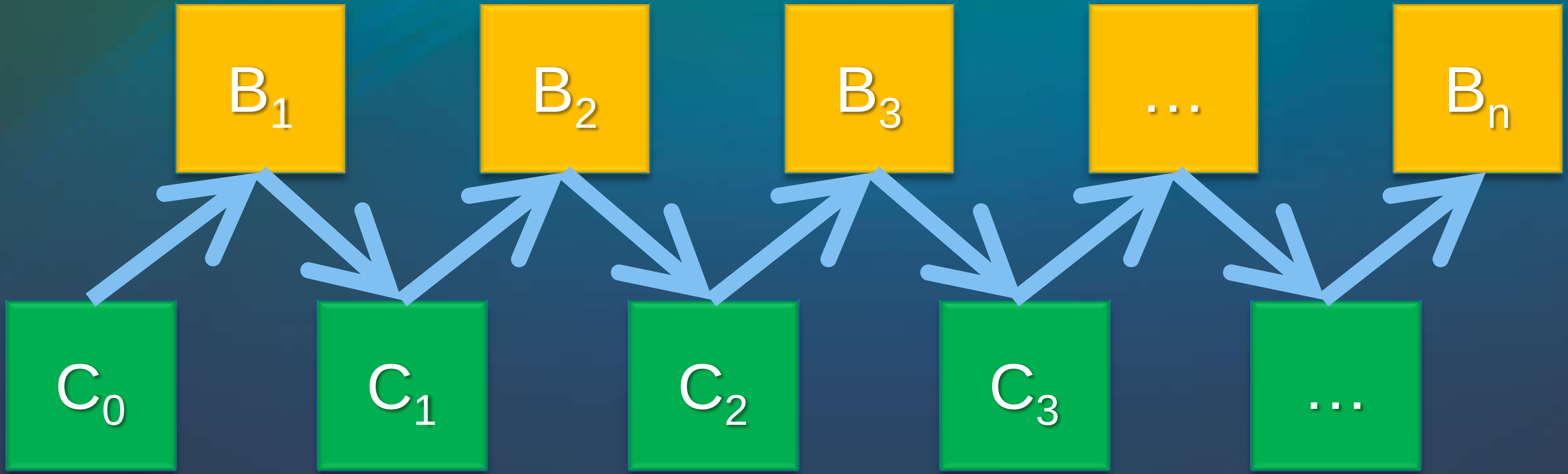
Fold (reduce)



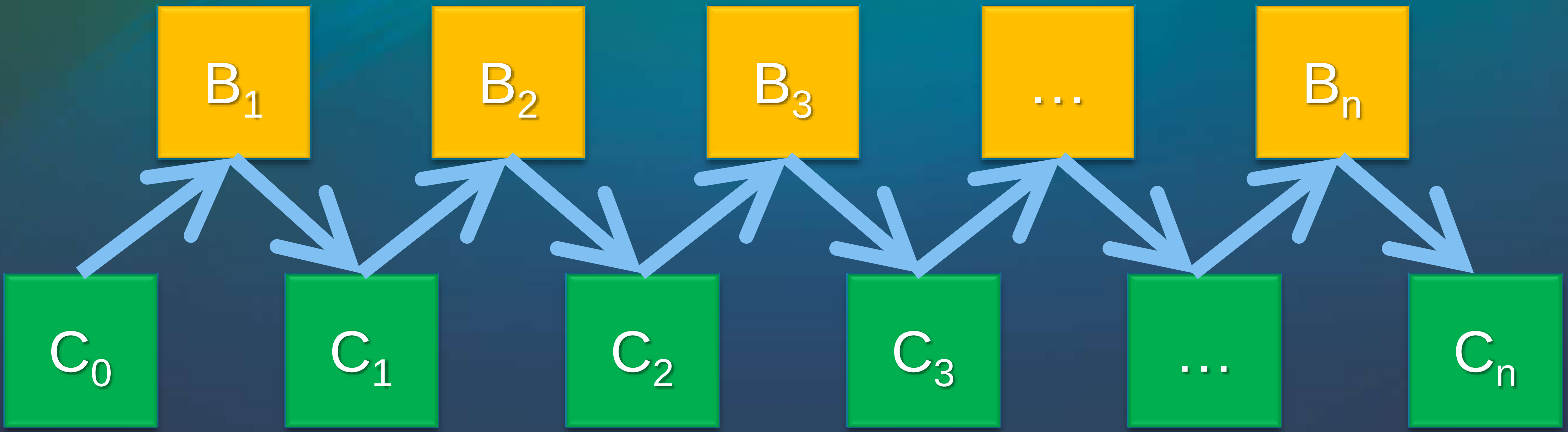
Fold (reduce)

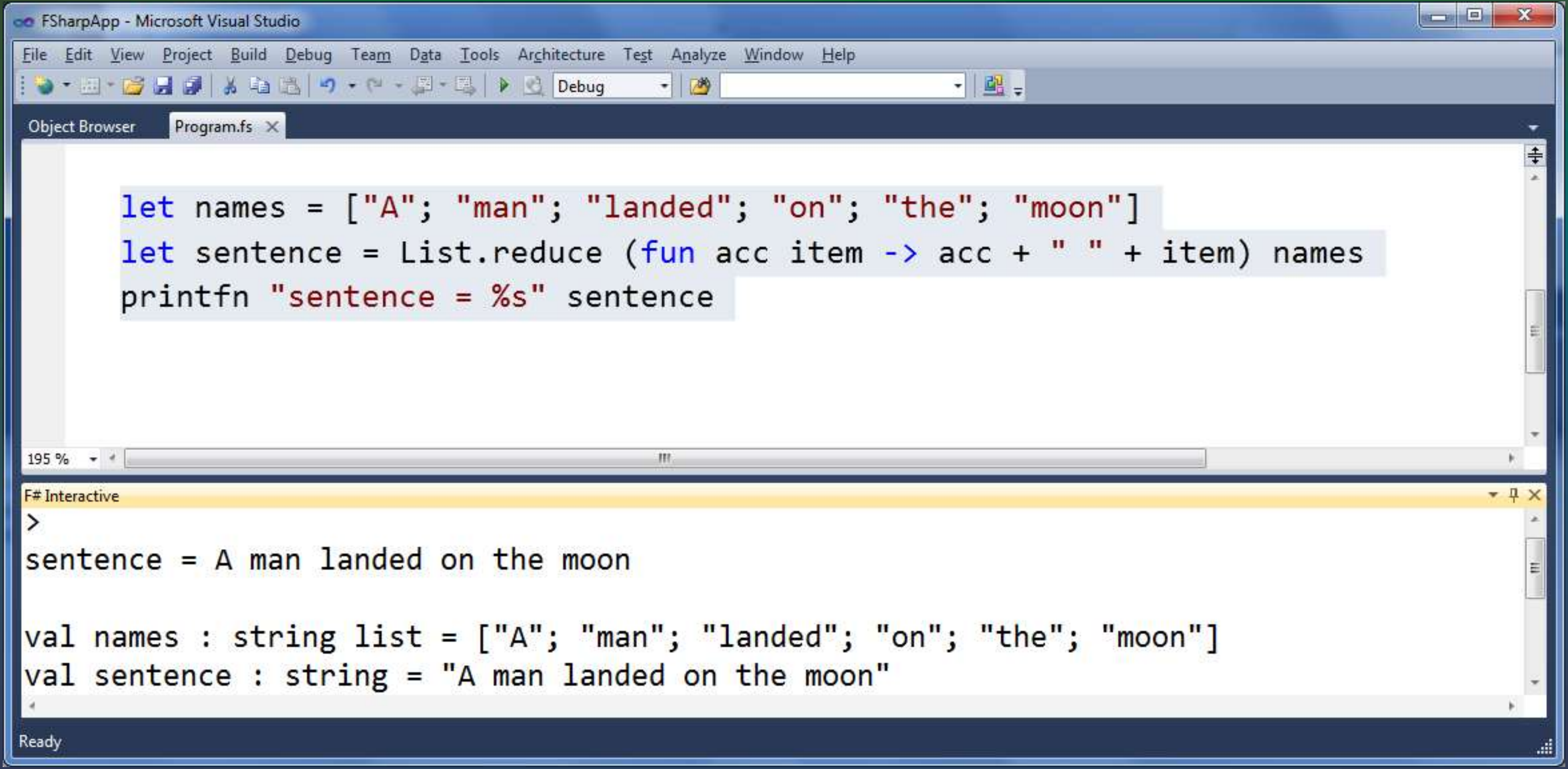


Fold (reduce)



Fold (reduce)

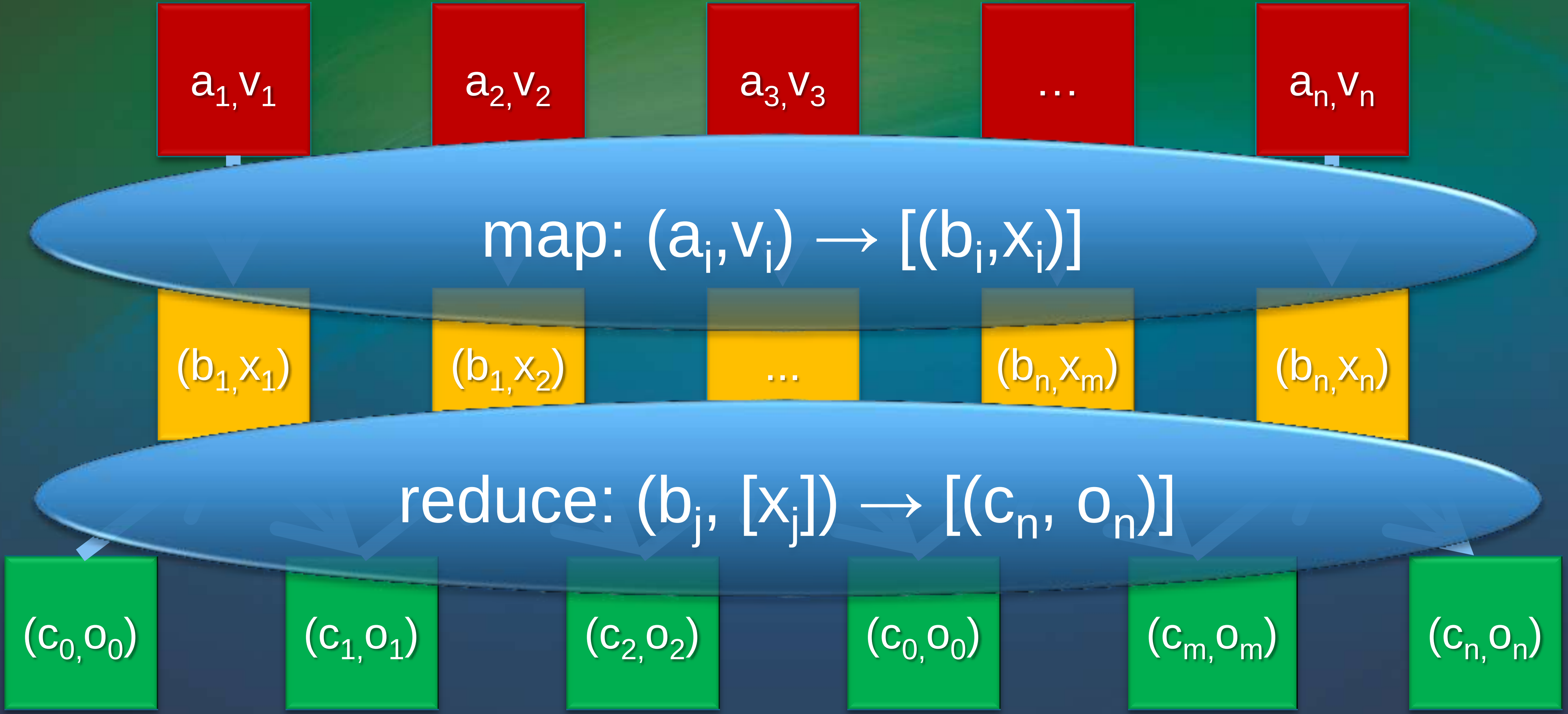



FSharpApp - Microsoft Visual Studio
File Edit View Project Build Debug Team Data Tools Architecture Test Analyze Window Help
Debug
Object Browser
Program.fs
195 %
F# Interactive
Ready

```
let names = ["A"; "man"; "landed"; "on"; "the"; "moon"]
let sentence = List.reduce (fun acc item -> acc + " " + item) names
printfn "sentence = %s" sentence
```

```
>
sentence = A man landed on the moon

val names : string list = ["A"; "man"; "landed"; "on"; "the"; "moon"]
val sentence : string = "A man landed on the moon"
```

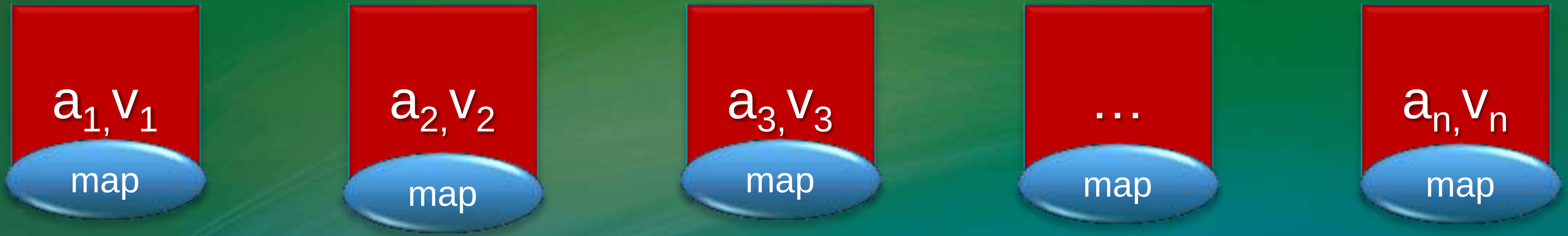
MapReduce as a framework



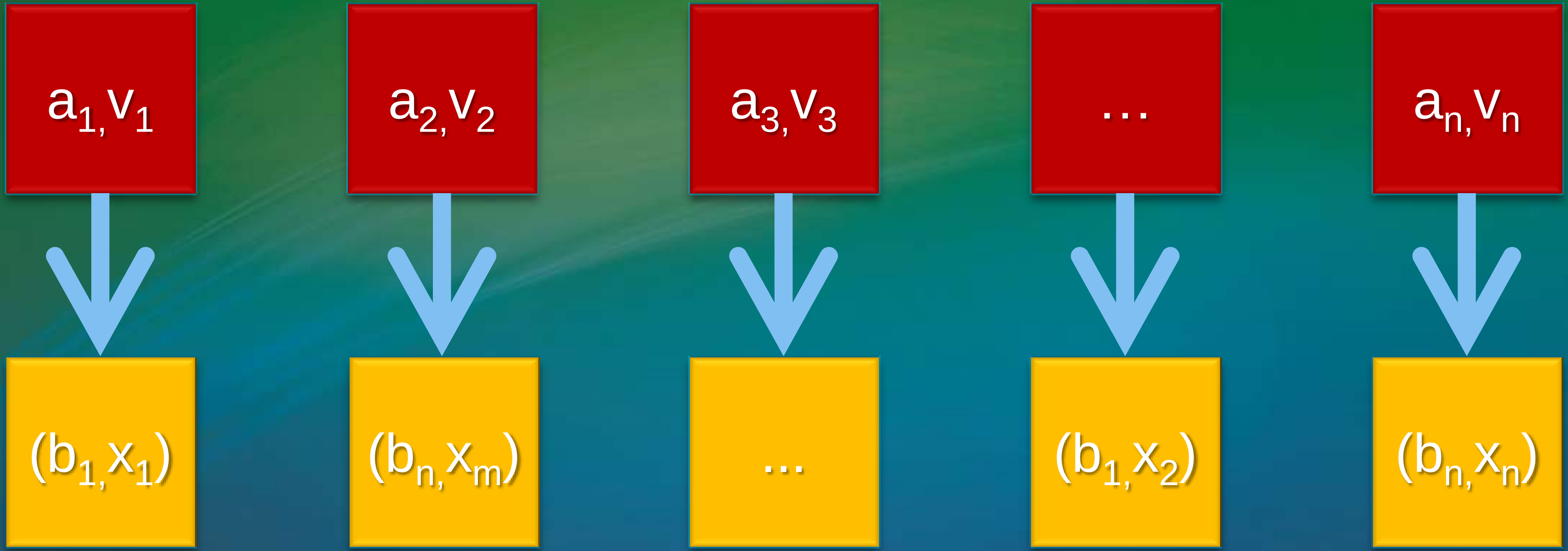
Framework execution

$a_{1,}v_1$ $a_{2,}v_2$ $a_{3,}v_3$ … $a_{n,}v_n$

Framework execution



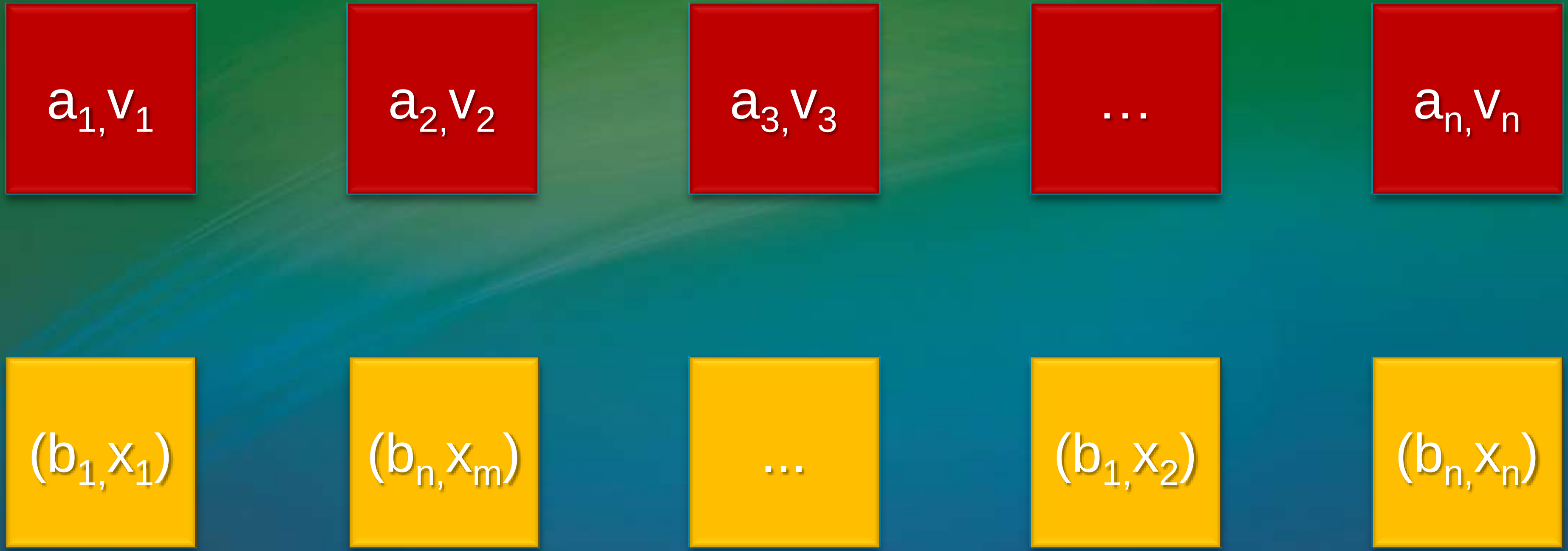
Framework execution



$a_{1,}v_1$ | $a_{2,}v_2$ | $a_{3,}v_3$ | … | $a_{n,}v_n$

$(b_{1,}x_1)$ | $(b_{n,}x_m)$ | … | $(b_{1,}x_2)$ | $(b_{n,}x_n)$

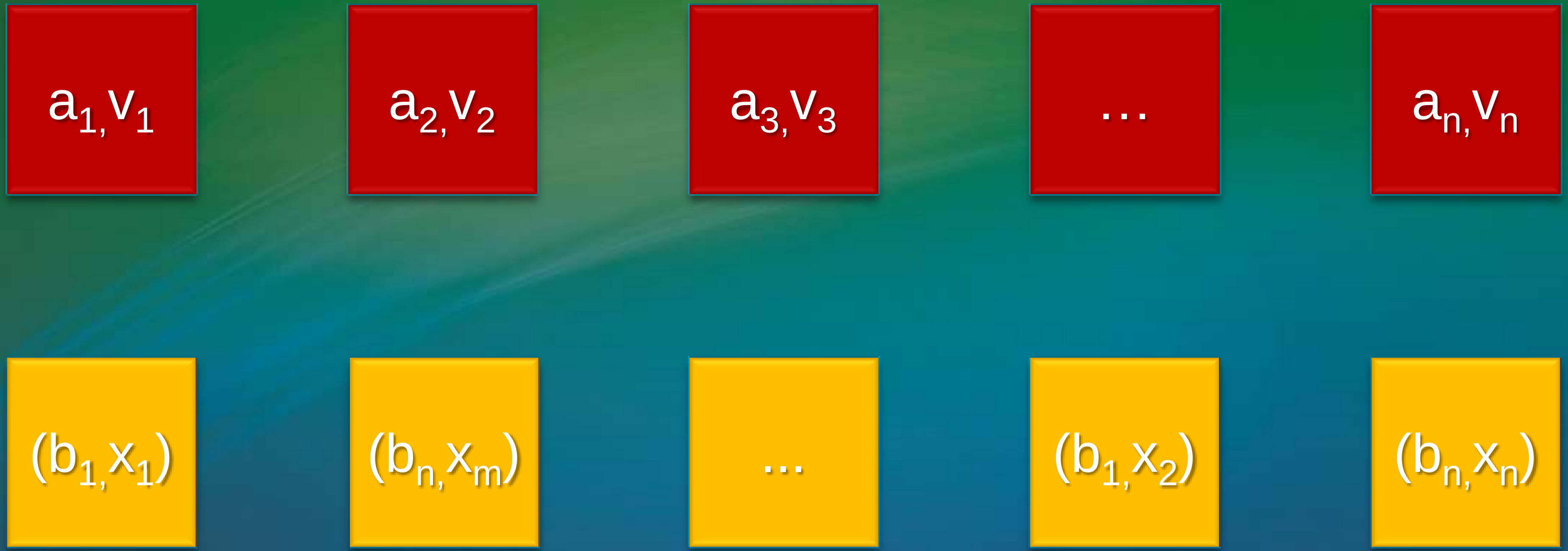
Framework execution



$a_{1,}v_1$ $a_{2,}v_2$ $a_{3,}v_3$ … $a_{n,}v_n$

$(b_{1,}x_1)$ $(b_{n,}x_m)$ … $(b_{1,}x_2)$ $(b_{n,}x_n)$

Framework execution



a_1, v_1

a_2, v_2

a_3, v_3

…

a_n, v_n

(b_1, x_1)

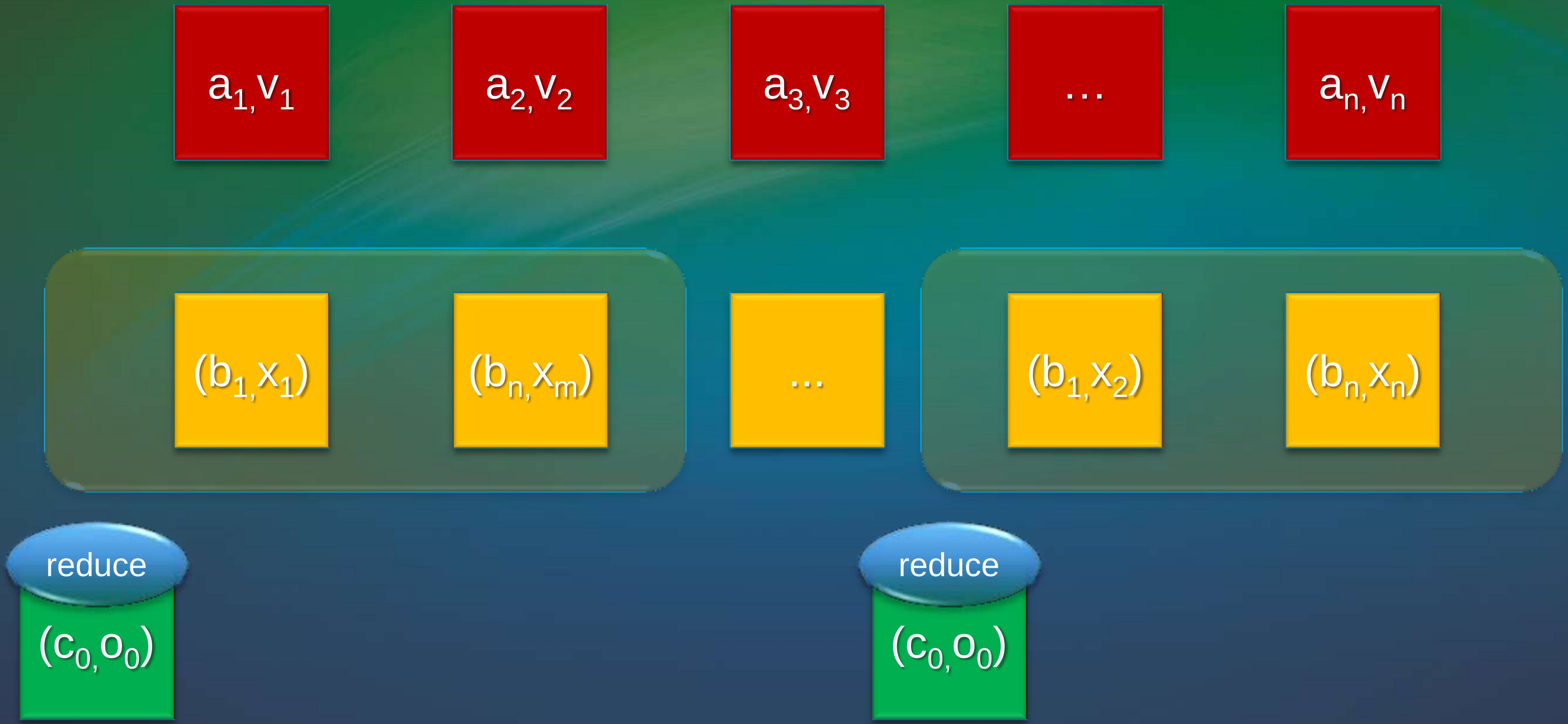
(b_n, x_m)

…

(b_1, x_2)

(b_n, x_n)

Framework execution



a_1, v_1

a_2, v_2

a_3, v_3

…

a_n, v_n

(b_1, x_1)

(b_n, x_m)

…

(b_1, x_2)

(b_n, x_n)

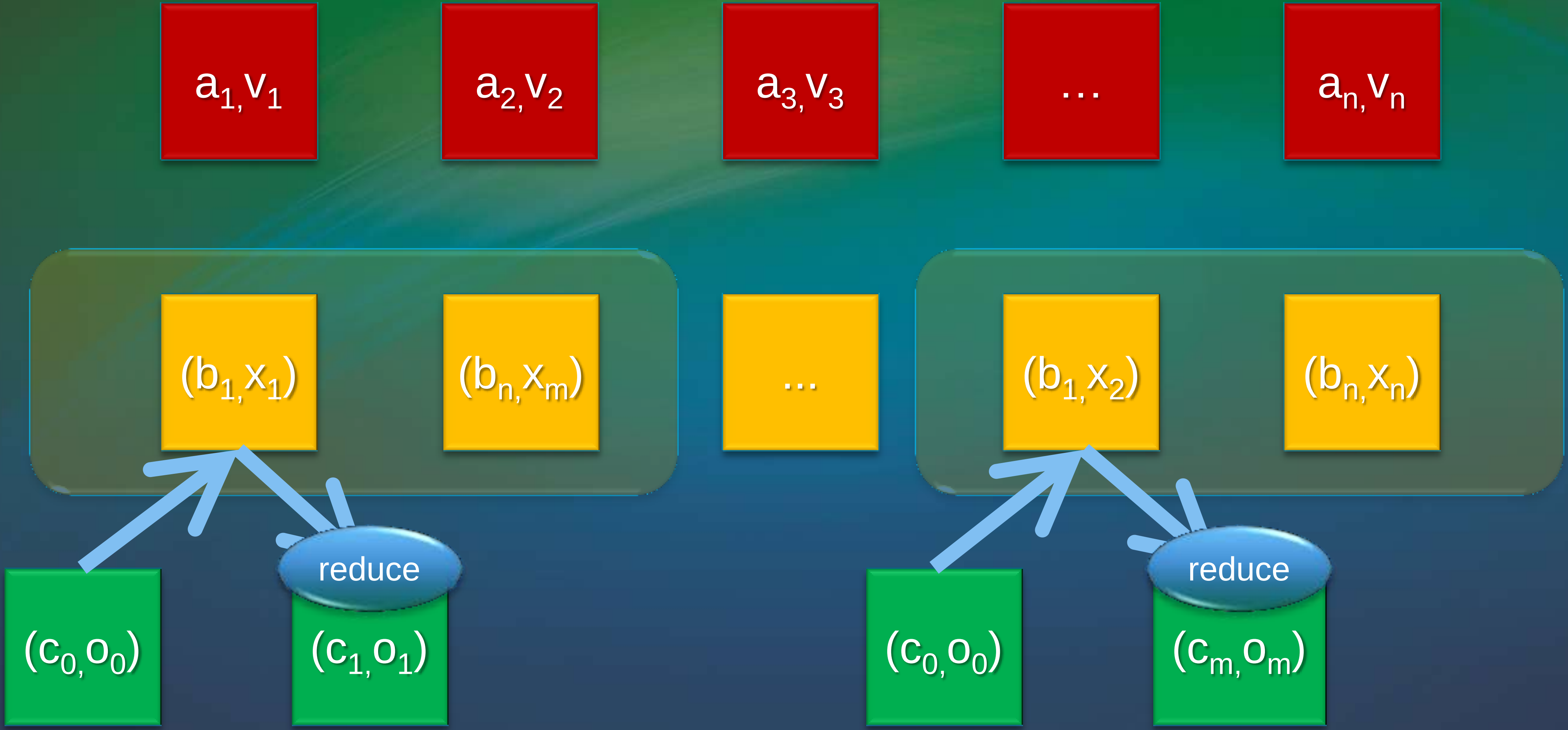
reduce

(c_0, o_0)

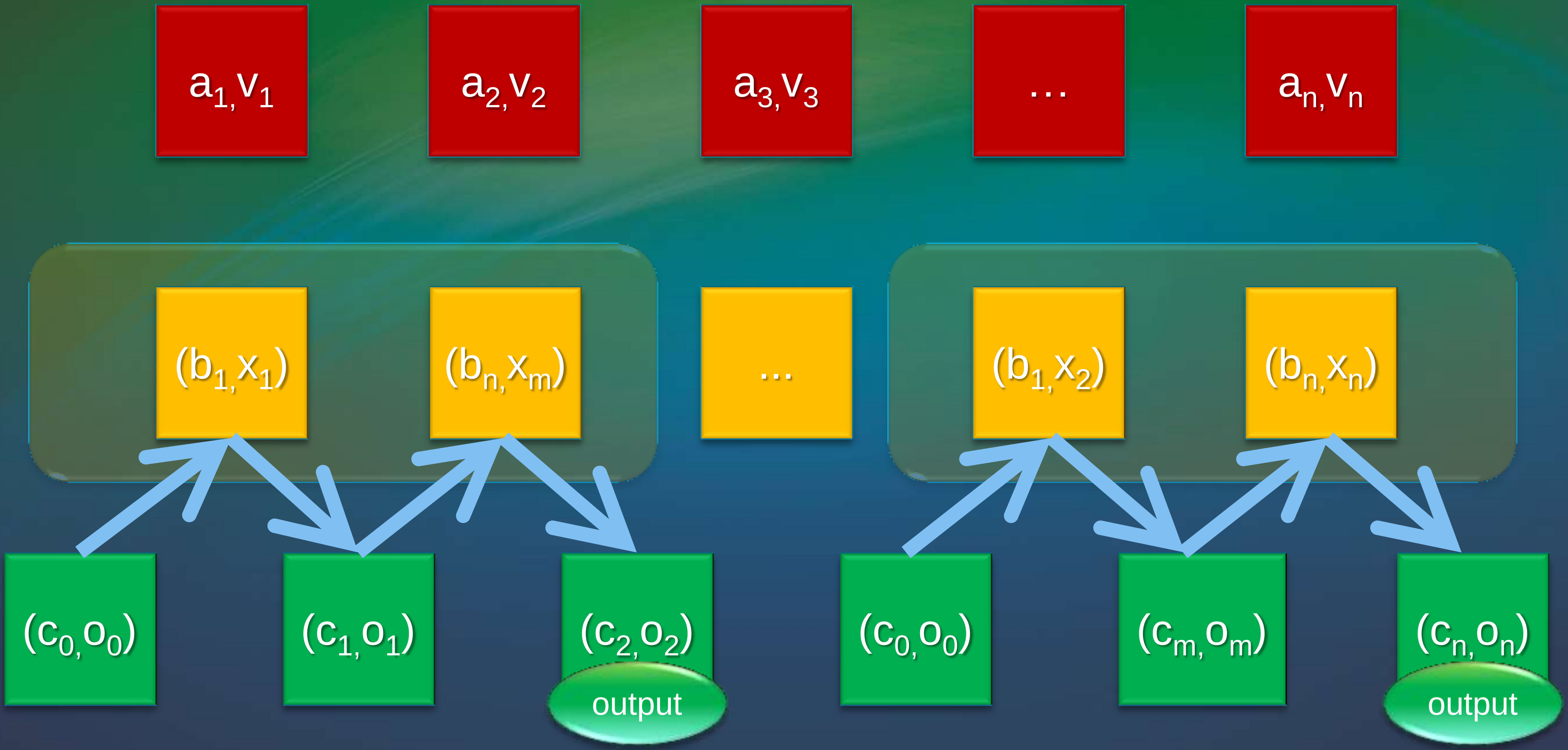
reduce

(c_0, o_0)

Framework execution



Framework execution



Count words in documents: Map

map: $(a_i,v_i) \rightarrow [(b_i,x_i)]$

```
map(docid a, string v):
  for each word b in v:
    Emit(string b, count 1)
```

Id	Text
K01	is here
K02	there

Id	Text
K11	there is
K12	is there

Id	Text
K21	from here
K22	from there

Id	Text
K31	is from
K32	here

Count words in documents: Map

map: $(a_i,v_i) \rightarrow [(b_i,x_i)]$

```
map(docid a, string v):
  for each word b in v:
    Emit(string b, count 1)
```

Id	Text
K01	is here
K02	there

Id	Text
K11	there is
K12	is there

Id	Text
K21	from here
K22	from there

Id	Text
K31	is from
K32	here

is	1
here	1
there	1

there	1
is	1
is	1
there	1

from	1
here	1
from	1
there	1

is	1
from	1
here	1

reduce: $(b_j, [x_j]) \rightarrow [(c_n, o_n))]$

Count words in documents: Reduce

```
reduce(string b, counts[$x_1$, $x_2$, …]):
    sum = 0
    for each x in counts:
        sum += x
    Emit(string b, int sum)
```

is	1
here	1
there	1

there	1
is	1
is	1
there	1

from	1
here	1
from	1
there	1

is	1
from	1
here	1

reduce: $(b_j, [x_j]) \rightarrow [(c_n, o_n))]$

Count words in documents: Reduce

```
reduce(string b, counts[$x_1$, $x_2$, …]):
   sum = 0
   for each x in counts:
      sum += x
   Emit(string b, int sum)
```

from	1
from	1
from	1

here	1
here	1
here	1

is	1
is	1
is	1
is	1

there	1
there	1
there	1
there	1

reduce: $(b_j, [x_j]) \rightarrow [(c_n, o_n))]$

Count words in documents: Reduce

```
reduce(string b, counts[$x_1$, $x_2$, …]):
  sum = 0
  for each x in counts:
    sum += x
  Emit(string b, int sum)
```

from	[1,1,1]
here	[1,1,1]
is	[1,1,1,1]
there	[1,1,1,1]

reduce: $(b_j, [x_j]) \rightarrow [(c_n, o_n))]$

Count words in documents: Reduce

```
reduce(string b, counts[$x_1$, $x_2$, …]):
  sum = 0
  for each x in counts:
    sum += x
  Emit(string b, int sum)
```

from	[1,1,1]	here	[1,1,1]	is	[1,1,1,1]	there	[1,1,1,1]
from	3	here	3	is	4	there	4

Approach and importance

- Large-scale data processing
- Cloud computing
  - Virtualization of the worker/slaves
  - Hosted Hadoop in Amazon Elastic MapReduce

Beyond MapReduce: LINQ to HPC

- Generates an execution plan based on a LINQ query
  - Supports more than just MapReduce!
- Runs on Windows HPC Server 2008 R2
  - HPC: High performance computing
- Integrated with Visual Studio 2010

Example: Find web pages from many log files

LINQ query transformed into computation graph

```
var logentries =
      from line in logs
      where !line.StartsWith("#")
      select new LogEntry(line);
var user =
      from access in logentries
      where access.user.EndsWith(@“Bob")
      select access;
var accesses =
      from access in user
      group access by access.page into pages
      select new UserPageCount(“Bob", pages.Key,    pages.Count());
var htmAccesses =
      from access in accesses
      where access.page.EndsWith(".htm")
      orderby access.count descending
      select access;
```

Example: Find web pages from many log files

LINQ query transformed into computation graph

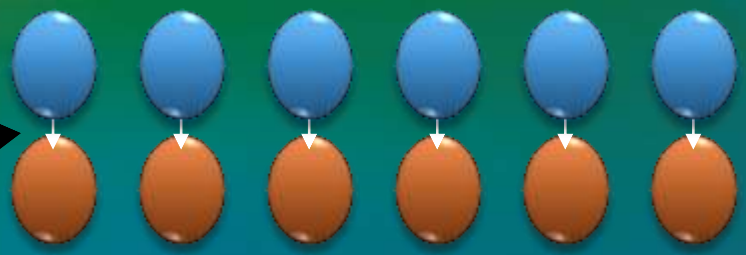
```
var logentries =
      from line in logs
      where !line.StartsWith("#")
      select new LogEntry(line);
var user =
      from access in logentries
      where access.user.EndsWith(@“Bob")
      select access;
var accesses =
      from access in user
      group access by access.page into pages
      select new UserPageCount(“Bob", pages.Key,    pages.Count());
var htmAccesses =
      from access in accesses
      where access.page.EndsWith(".htm")
      orderby access.count descending
      select access;
```

Example: Find web pages from many log files

LINQ query transformed into computation graph

```
var logentries =
      from line in logs
      where !line.StartsWith("#")
      select new LogEntry(line);
var user =
      from access in logentries
      where access.user.EndsWith(@“Bob")
      select access;
var accesses =
      from access in user
      group access by access.page into pages
      select new UserPageCount(“Bob", pages.Key,    pages.Count());
var htmAccesses =
      from access in accesses
      where access.page.EndsWith(".htm")
      orderby access.count descending
      select access;
```

1 Input



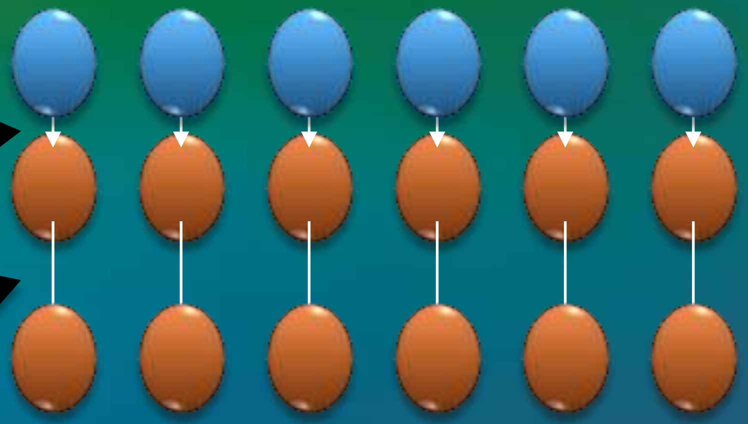
Example: Find web pages from many log files

LINQ query transformed into computation graph

```
var logentries =
      from line in logs
      where !line.StartsWith("#")
      select new LogEntry(line);
var user =
      from access in logentries
      where access.user.EndsWith(@“Bob")
      select access;
var accesses =
      from access in user
      group access by access.page into pages
      select new UserPageCount(“Bob", pages.Key,    pages.Count());
var htmAccesses =
      from access in accesses
      where access.page.EndsWith(".htm")
      orderby access.count descending
      select access;
```

1 Input

2 Compute



Example: Find web pages from many log files

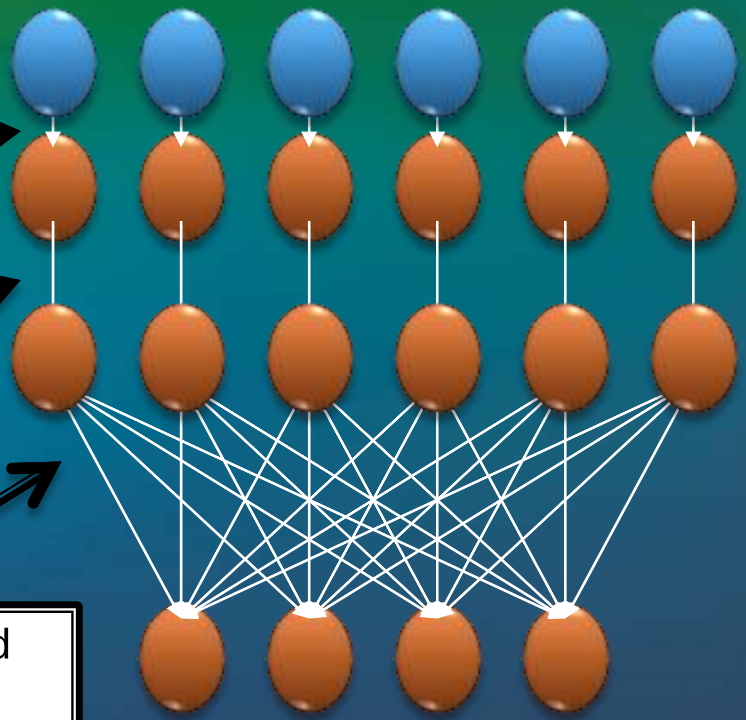
LINQ query transformed into computation graph

```
var logentries =
      from line in logs
      where !line.StartsWith("#")
      select new LogEntry(line);
var user =
      from access in logentries
      where access.user.EndsWith(@“Bob")
      select access;
var accesses =
      from access in user
      group access by access.page into pages
      select new UserPageCount(“Bob", pages.Key,    pages.Count());
var htmAccesses =
      from access in accesses
      where access.page.EndsWith(".htm")
      orderby access.count descending
      select access;
```

1 Input

Compute 2

Compute and resort



Example: Find web pages from many log files

LINQ query transformed into computation graph

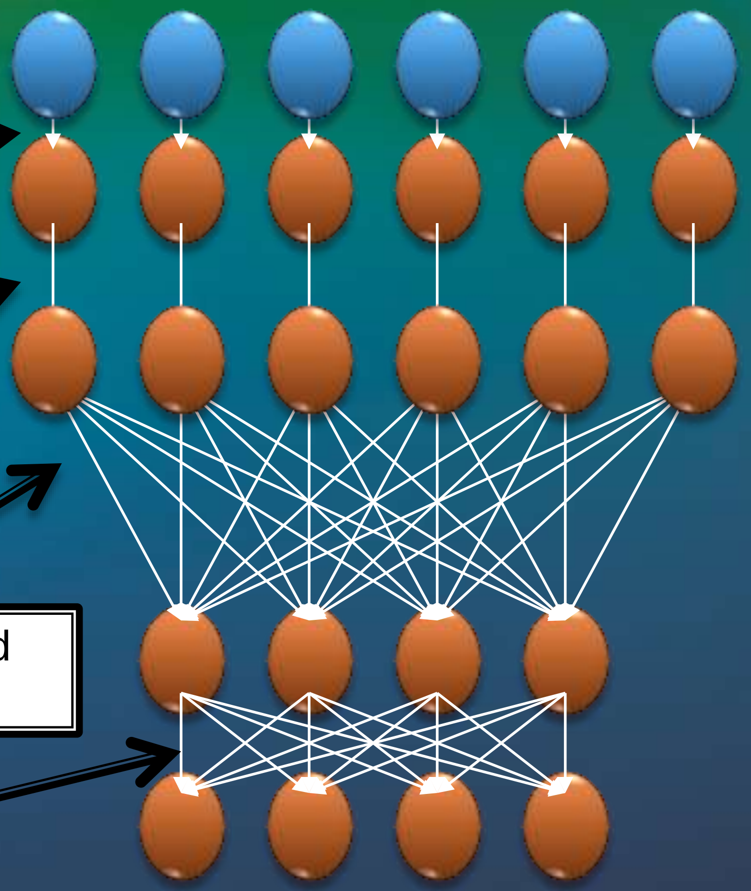
```
var logentries =
      from line in logs
      where !line.StartsWith("#")
      select new LogEntry(line);
var user =
      from access in logentries
      where access.user.EndsWith(@“Bob")
      select access;
var accesses =
      from access in user
      group access by access.page into pages
      select new UserPageCount(“Bob", pages.Key,    pages.Count());
var htmAccesses =
      from access in accesses
      where access.page.EndsWith(".htm")
      orderby access.count descending
      select access;
```

1 Input

2 Compute

● Compute and resort

4 Compute and resort



Example: Find web pages from many log files

LINQ query transformed into computation graph

```
var logentries =
      from line in logs
      where !line.StartsWith("#")
      select new LogEntry(line);
var user =
      from access in logentries
      where access.user.EndsWith(@“Bob")
      select access;
var accesses =
      from access in user
      group access by access.page into pages
      select new UserPageCount(“Bob", pages.Key,    pages.Count());
var htmAccesses =
      from access in accesses
      where access.page.EndsWith(".htm")
      orderby access.count descending
      select access;
```

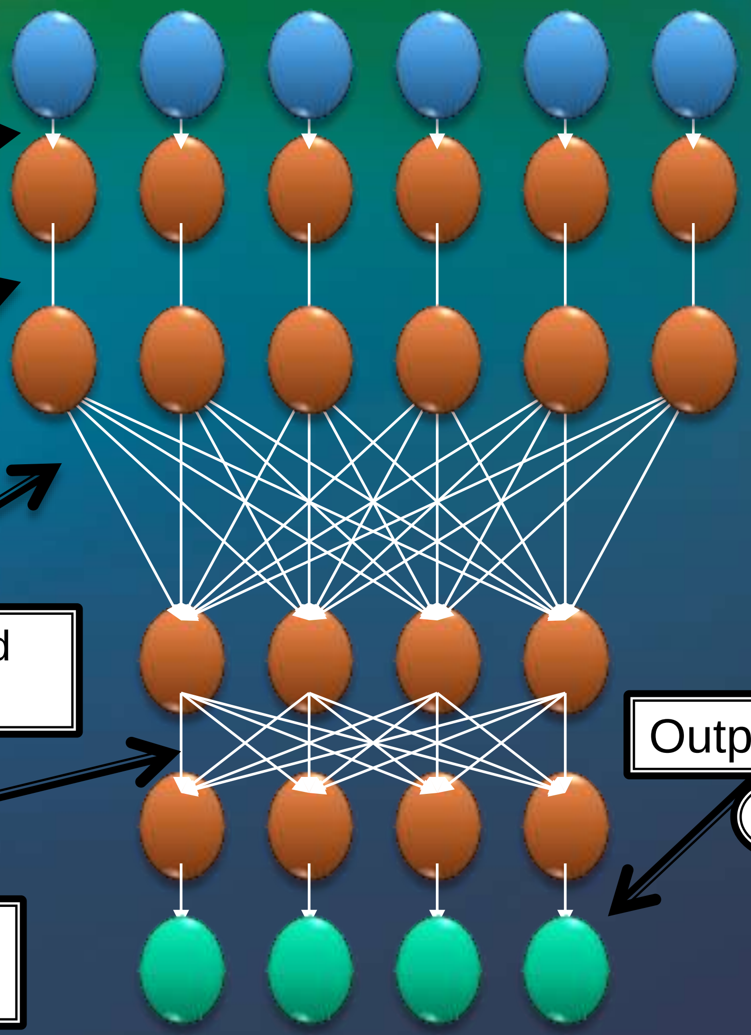
1 Input

2 Compute

Compute and resort

4 Compute and resort

5 Output



Word count in LINQ to HPC

```
public static IQueryable<Pair> Histogram(IQueryable<string> input)
{
  IQueryable<string> words = input.SelectMany(x => x.Split(' '));
  IQueryable<IQueryable<string,string>> groups = words.GroupBy(x => x);
  IQueryable<Pair> counts = groups.Select(x => new Pair(x.Key, x.Count()));

  return counts;
}
```

```
public struct Pair {
    string word;
    int count;
    public Pair(string w, int c)
    {
        word = w;
        count = c;
    }
    public override string string ToString() {
        return word + ":" + count.ToString();
    }
}
```

Call to action

- Architect your systems using message queues
  - Windows Communication Foundation
  - Azure AppFabric Service Bus
- Learn more about functional programming
  - Play with F#: http://www.fsharp.net
- Learn more about LINQ
- Download and play with LINQ to HPC
  - http://www.microsoft.com/hpc (Windows HPC Pack)
  - http://connect.microsoft.com/hpc (samples and documentation)

References

- *Data-Intensive Text Processing with MapReduce*
  by Jimmy Lin, Chris Dyer
- *Brewer's Conjecture and the Feasibility of the Consistent, Available, Partition-Tolerant Web Services*
  by Seth Gilbert, Nancy Lynch
  Document link
- *Architecting for Latency*
  by Dan Pritchett
  Colorado Software Summit, 2007
  Presentation link

Acknowledgements

- Thanks to all the reviewers
- Thanks to Ade Miller (HPC-PM) for sharing content about LINQ to HPC
- Thanks to Sir Tony Hoare for sharing insights into the future of programming

How will we be programming in 100 years?

Extrapolating current trends, the job of programming will consist in making tiny local changes to the vast corpus of legacy software that runs on the world-wide computer network, measured in terabytes of code

—Tony Hoare