

Tema 4. Análisis comparativo del rendimiento

¿Qué servidor tiene mejor rendimiento?

Analistas, administradores y diseñadores



Objetivos del tema

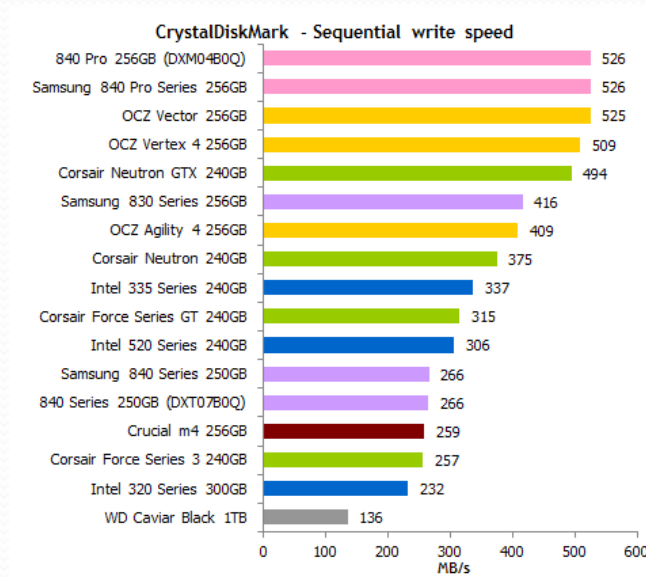
- Entender la problemática inherente al diseño de un índice de rendimiento cualquiera.
- Interpretar los índices clásicos de rendimiento usados en el ámbito de los procesadores.
- Entender el concepto de benchmark y sus distintos tipos.
- Conocer ejemplos reales de benchmarks.
- Conocer diferentes estrategias de análisis para hacer comparaciones de rendimiento así como las condiciones para hacer una comparación de rendimiento lo más ecuánime posible.

Bibliografía

- *Evaluación y modelado del rendimiento de los sistemas informáticos*. Xavier Molero, C. Juiz, M. Rodeño. Pearson Educación, 2004. Capítulo 3.
- *Measuring computer performance: a practitioner's guide*. David J. Lilja, Cambridge University Press, 2000. Capítulos 2,5 y 7.
- *The art of computer systems performance analysis : Techniques for experimental design, measurement, simulation, and modeling*. Raj Jain, John Wiley & Sons, 1991. Capítulos 9, 13 y 20.
- *System Performance Tuning*. Gian-Paolo D. Musumeci, Mike Loukides, 2nd Edition - O'Reilly Media, 2002. Capítulo 2.
- *The Standard Performance Evaluation Corporation (SPEC)*, <http://www.spec.org>.
- *The Transaction Processing Performance Council (TPC)*, <http://www.tpc.org>.

Contenido

- Referenciación (benchmarking).
- Análisis de los resultados de un test de rendimiento.
- Comparación de prestaciones en presencia de aleatoriedad.
- Diseño de experimentos de comparación de rendimiento.



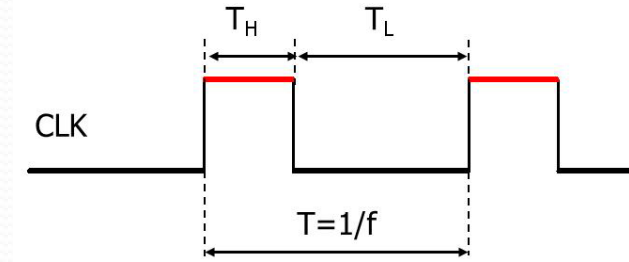
4.1. Referenciación (*Benchmarking*)

Características de un buen índice de rendimiento de un sistema informático

- **Repetibilidad:** **Siempre** que se mida el índice en las mismas condiciones, el valor de éste debe ser el mismo.
- **Representatividad y fiabilidad:** Si un sistema A **siempre** presenta un índice de rendimiento mejor que el sistema B, es porque **siempre** el rendimiento real de A es mejor que el de B.
- **Consistencia:** El índice se debe poder medir en cualquier sistema informático, independientemente de su arquitectura o de su S.O.
- **Facilidad de medición:** Sería deseable que las medidas necesarias para obtener el índice sean fáciles de tomar.
- **Linealidad:** Sería deseable que, si el índice de rendimiento aumenta, el rendimiento real del sistema aumente en la misma proporción.

Tiempo de ejecución, frecuencia de reloj y CPI

¿Pueden ser la frecuencia de reloj (f_{RELOJ}) o el número medio de ciclos por instrucción (**CPI**) buenos índices de rendimiento?



$$T_{EJEC} = NI \times CPI \times T_{RELOJ} = \frac{NI \times CPI}{f_{RELOJ}}$$

- T_{EJEC} = Tiempo de ejecución del programa.
- NI = Número de instrucciones del programa.

- No lo son. Es posible encontrar ejemplos de sistemas con f_{RELOJ} (o CPI) peores que otros pero con mejores prestaciones. Además, CPI depende del programa que se ejecute.
- ¿Y si usamos directamente el tiempo de ejecución (T_{EJEC}) **de un determinado programa**?
 - ¿Consistencia? El programa debería estar descrito en un lenguaje de alto nivel. Ok.
 - ¿Repetibilidad? El programa debería ejecutarse en un entorno muy controlado. Ok.
 - ¿Representatividad y fiabilidad? **¡¡¡Dependería del programa a ejecutar!!!**

MIPS (million of instructions per second)

- En principio, parece una medida prometedora ya que representa cómo de rápido ejecuta las instrucciones un microprocesador.

$$MIPS = \frac{NI}{T_{EJEC} \times 10^6} = \frac{f_{RELOJ}}{CPI \times 10^6}$$

Inconvenientes:

- ¿Repetibilidad? Los MIPS medidos varían incluso entre diferentes programas en el mismo computador.
- ¿Representatividad y fiabilidad? Depende del juego de instrucciones (ej. RISC vs CISC).

```
def add_forty_two(n)

    pushq %rbp
    movq  %rsp, %rbp
    addl  $42, %edi
    movl  %edi, %eax
    popq  %rbp
    retq

end
```

```
slli x30, x5, 3 // x30 = f*8
add x30, x10, x30 // x30 = &A[f]
slli x31, x6, 3 // x31 = g*8
add x31, x11, x31 // x31 = &B[g]
ld x5, 0(x30) // f = A[f]
addi x12, x30, 8
ld x30, 0(x12)
add x30, x30, x5
sd x30, 0(x31)
```

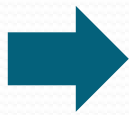

MFLOPS (*million of floating-point operations per second*)

- Basado en operaciones y no en instrucciones.

$$MFLOPS = \frac{\text{Operaciones de coma flotante realizadas}}{T_{EJEC} \times 10^6}$$

Inconvenientes:

- No todas las operaciones de coma flotante tienen la misma complejidad. Posible solución: **MFLOPS normalizados**: Cada operación se multiplica por un peso que es proporcional a su complejidad. Ejemplo de asignación de pesos:
 - ADD, SUB, COMPARE, MULT \Rightarrow 1 operación normalizada
 - DIVIDE, SQRT \Rightarrow 4 operaciones normalizadas
 - EXP, SIN, ATAN, ... \Rightarrow 8 operaciones normalizadas
- ¿Consistencia? El formato de los números en coma flotante puede variar de una arquitectura a otra y, por tanto, los resultados de las operaciones podrían tener diferente exactitud.
- Además, ¿y si no son importantes las operaciones en coma flotante en mi servidor?

 **Conclusión final:** Tampoco nos vale y no hay más candidatos. Nos contentaremos con el tiempo de ejecución (T_{EJEC}) de un determinado programa o conjunto de programas \rightarrow **El índice de rendimiento va a depender de la carga que se use para hacer la comparación.**

La carga real

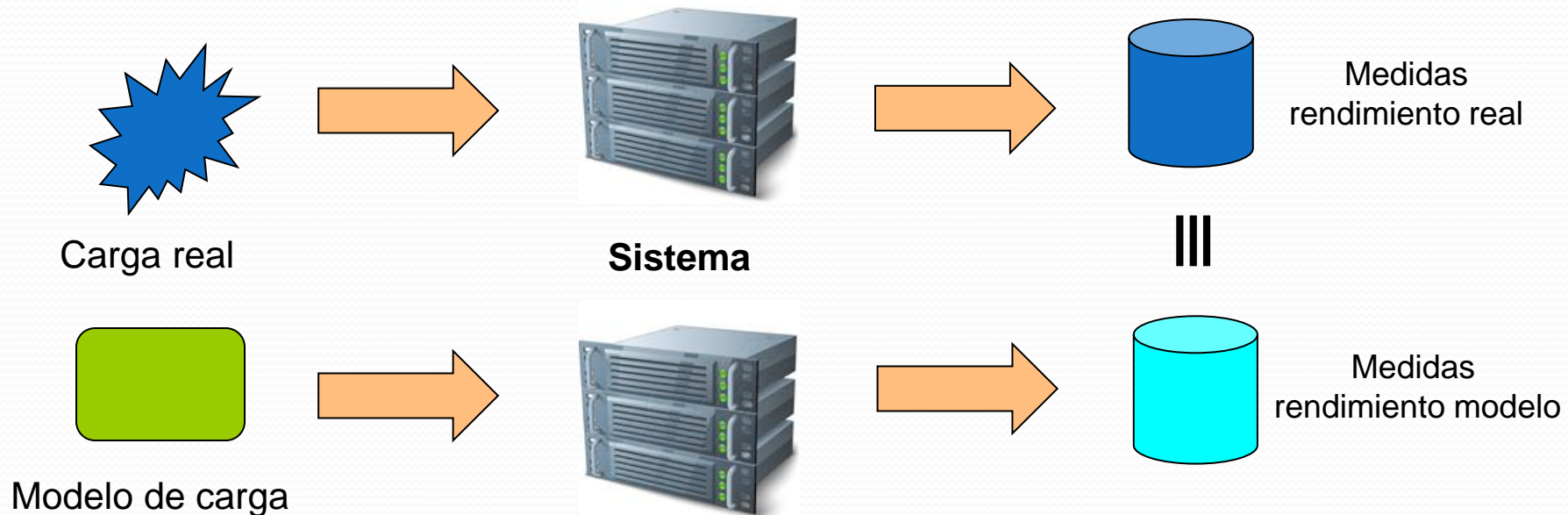
- Difícil de utilizar para la comparación de rendimiento de un servidor por las siguientes razones:
 - Varía a lo largo del tiempo: ¿Uso la carga de hoy, la de ayer...?
 - Resulta complicado reproducirla: Si la uso con un servidor muy lento, ¿es posible que llegue una solicitud basada en una respuesta que todavía no ha llegado!!
 - Interacciona con el sistema informático: Si un servidor contesta muy rápido, es posible que reciba las solicitudes antes o incluso más solicitudes que si fuera muy lento.



- Es más conveniente utilizar un **modelo** de la carga real como carga de prueba (*test workload*) para hacer comparaciones.

Representatividad del modelo de carga

- Los modelos de carga son representaciones aproximadas de la carga que recibe un sistema informático. El modelo de la carga:
 - Debe ser lo más representativo posible de la carga real.
 - Debe ser lo más simple/compacto que sea posible (tiempos de medición y espacio en memoria razonables).



Principales estrategias para obtener modelos de carga



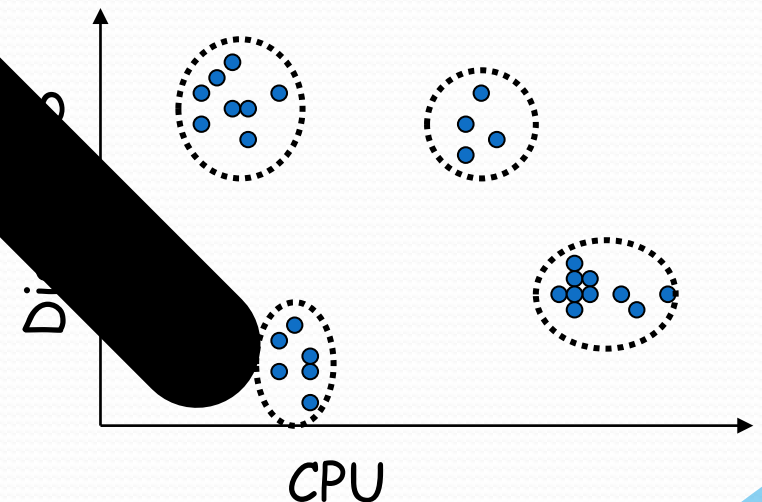
- Ajustar un modelo paramétrico “**personalizado**” a partir de la monitorización del sistema ante la carga real (*caracterización de la carga*).



- Usar cargas de prueba que usen un modelo **genérico** de carga lo más similar posible al que se quiere reproducir (*referenciación o benchmarking*).

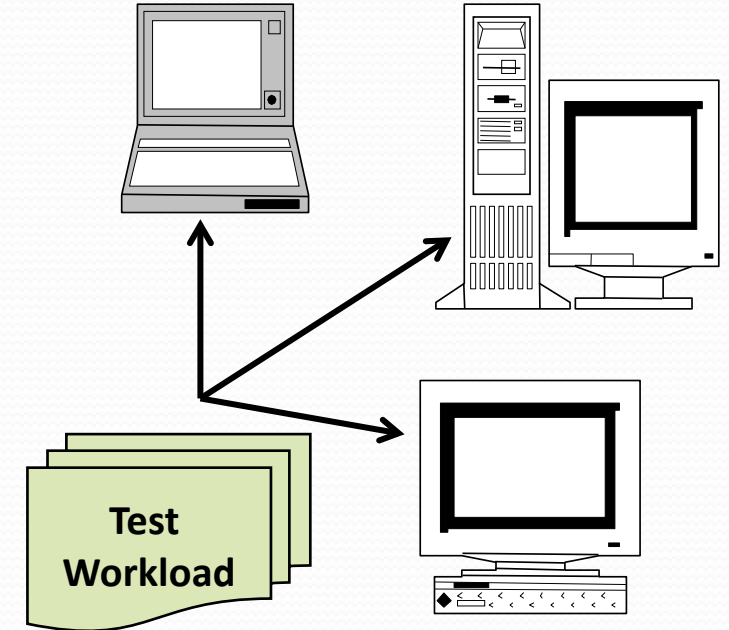
Caracterización de la carga

- La forma más fácil para obtener un modelo de la carga a la que está sometido un servidor durante un determinado periodo de tiempo consiste en:
 - Identificar los recursos que maneja la carga (CPU, memoria, discos, red, etc.)
 - Elegir los parámetros característicos de dichos recursos (utilización de CPU, lecturas/escrituras que hay que hacer en cada disco, lecturas/escrituras a memoria, número de accesos a la red, etc.)
 - Medir el valor de dichos parámetros usando algún tipo de actividad (muestreo).
 - Analizar los datos: medias, histogramas, agrupamiento o *clustering*, etc.
 - Generar el modelo de carga seleccionando los *tipos de la carga* (=solicitudes al servidor) junto con información estadística sobre su distribución temporal.
- Ventajas:
 - La carga de prueba es muy representativa de la carga real.
- Desventajas:
 - El proceso de obtención de la carga de prueba es bastante tedioso.
 - Desconocemos el rendimiento de otros servidores distintos al nuestro con esa carga de prueba.



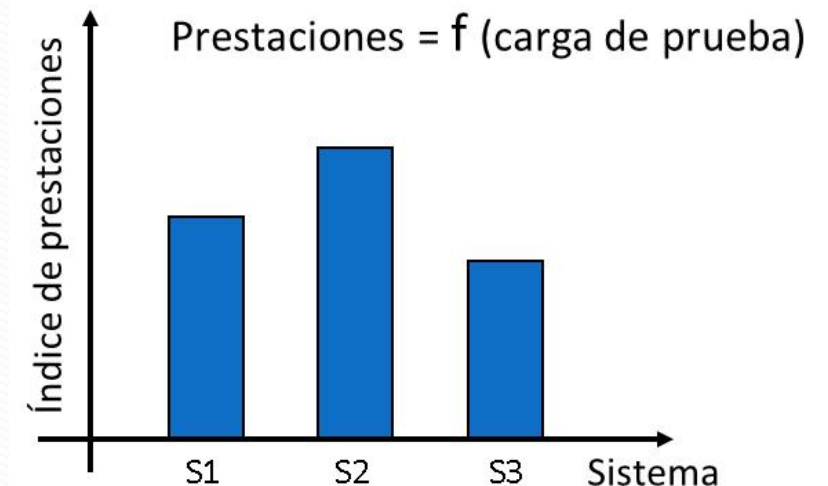
Referenciación (*Benchmarking*)

- Consiste en utilizar un programa o un conjunto de programas (*benchmark programs*) estándar con el fin de comparar alguna característica del rendimiento (también llamada “*métrica*”) entre equipos informáticos. Hay dos características principales que definen a un *benchmark*:
 - La **carga de prueba** (*test workload*) específica con la que estresa el sistema evaluado.
 - El **conjunto de reglas** que se deben seguir para la correcta ejecución, obtención y validación de los resultados.



Ventajas de usar *benchmarking*

- Existen muchos *benchmarks* diferentes para distintos tipos de servidores y cargas. Hay una **alta probabilidad** de encontrar uno que reproduzca unas condiciones parecidas a las que experimenta nuestro servidor.
- Las comparaciones entre el rendimiento de varios servidores son **justas** ya que todas las ejecuciones se realizan de forma idéntica siguiendo las **reglas del *benchmark***.
- Muchos *benchmarks* permiten ajustar la carga de tal forma que podemos medir la **escalabilidad** de nuestro servidor.
- Al poder conocer tanto el rendimiento para un determinado *benchmark* que obtienen diferentes servidores como cómo están diseñados y configurados dichos servidores, obtenemos una **información muy valiosa sobre cómo diseñar y/o configurar nuestros propios servidores**.



Tipos de programas de benchmark: según la estrategia de medida

- Programas que miden el tiempo necesario para ejecutar una cantidad pre-establecida de tareas.
 - La mayoría de benchmarks.
- Programas que miden la cantidad de tareas ejecutadas para un tiempo de cómputo pre-establecido.
 - SLALOM: Mide la exactitud de la solución de un determinado problema que se puede alcanzar en 1 minuto de ejecución.
 - TPC-C: Calcula cuántas consultas por segundo se realizan, de media, a un servidor de base de datos permitiendo aumentar tanto el nº de usuarios como el tamaño de la base de datos. Exige un tiempo mínimo de respuesta para un determinado tanto por ciento de peticiones.

Tipos de programas de benchmark: según la generalidad del test

- Microbenchmarks o benchmarks para **componentes**: estresan componentes o agrupaciones de componentes concretos del sistema: procesador, caché, memoria, discos, red, procesador+caché, procesador+compilador+memoria virtual, etc.
- Macrobenchmarks o benchmarks de sistema **completo** o de **aplicación real**: la carga intenta imitar situaciones reales (normalmente servidores con muchos clientes) típicas de algún área. P.ej. comercio electrónico, servidores web, servidores de ficheros, servidores de bases de datos, sistemas de ayuda a la decisión, paquetes ofimáticos + correo electrónico + navegación, etc.



Ejemplos de microbenchmarks

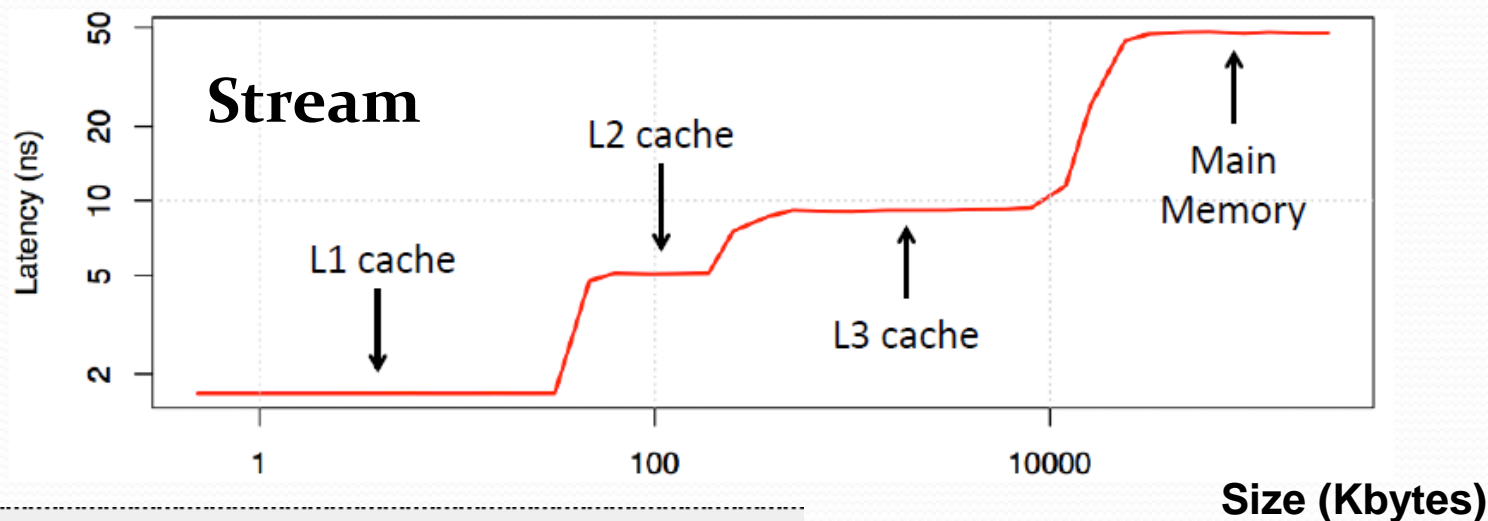
- *Whetstone* (1976)
 - Mide el rendimiento de las operaciones en **coma flotante** por medio de pequeñas aplicaciones científicas que usan sumas, multiplicaciones y funciones trigonométricas.
- *Linpac* (1983)
 - Mide el rendimiento de las operaciones en **coma flotante** a través de un algoritmo para resolver un sistema denso de ecuaciones lineales. El benchmark incorpora una rutina para comprobar que la solución a la que se llega es la correcta con un grado de exactitud prefijado. Se utiliza para confeccionar la lista de los [500 mejores supercomputadores del mundo](#).
- *Dhrystone* (1984)
 - Mide el rendimiento de operaciones con **enteros**, esencialmente por medio de operaciones de copia y comparación de cadenas de caracteres.

Ejemplos de microbenchmarks (II)

- *Stream*: para medir el ancho de banda de la memoria DRAM y las cachés del microprocesador (SRAM) <https://github.com/jeffhammond/STREAM>.
- *IOzone*: rendimiento del sistema de ficheros (p.ej. lecturas y escrituras a/desde el disco duro), <http://www.iozone.org/>. Igualmente *HD Tune* (Windows, <http://www.hdtune.com/>), *Iometer* (<http://www.iometer.org/>), *fio* (Linux, <https://fio.readthedocs.io/>, flexible I/O tester,) o el programa '*hdparm -tT*' (Linux).
- *Iperf*: rendimiento TCP y UDP (Linux y Windows). Debe estar instalado tanto en el cliente como en el servidor (<https://iperf.fr/>). O el programa *pchar* (Linux), calcula el ancho de banda por cada salto de IP hasta alcanzar el destino.
- También existen aplicaciones que incorporan varios **paquetes de microbenchmarks** para poder realizar diversos tests de forma cómoda:
 - *AIDA64* (Windows, <http://www.aida64.com>).
 - *Sandra* (Windows, <http://www.sisoftware.net>).
 - *Phoronix Test Suite* (Open Source, <https://www.phoronix-test-suite.com/>).



Ejemplos de micro-benchmarks (III)



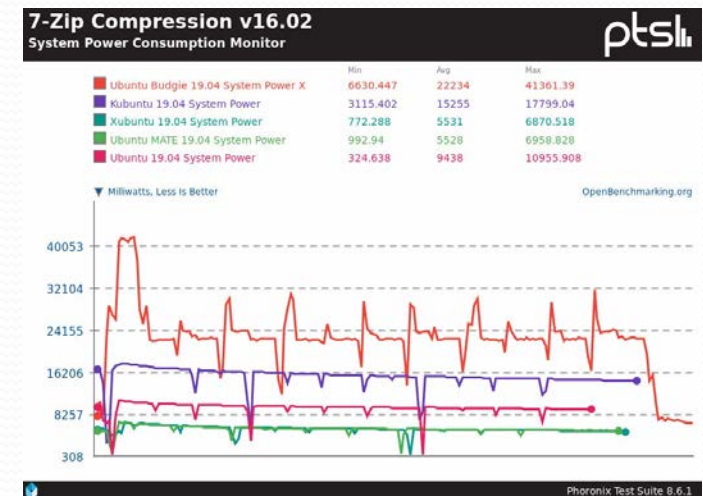
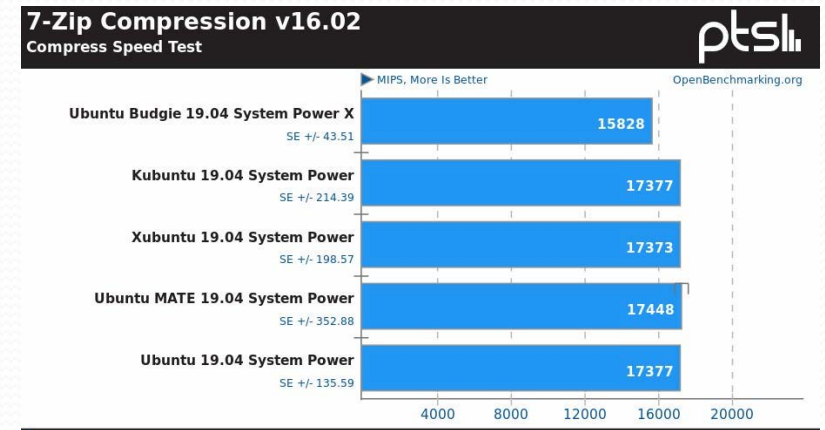
```
$ fio --name=seqwrite --rw=write --bs=128k --size=122374m
[...]
seqwrite: (groupid=0, jobs=1): err= 0: pid=22321
  write: io=122374MB, bw=840951KB/s, iops=6569 , runt=149011msec
    clat (usec): min=41 , max=133186 , avg=148.26, stdev=1287.17
    lat (usec): min=44 , max=133188 , avg=151.11, stdev=1287.21
    bw (KB/s) : min=10746, max=1983488, per=100.18%, avg=842503.94,
stdev=262774.35
  cpu           : usr=2.67%, sys=43.46%, ctx=14284, majf=1, minf=24
  IO depths     : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
  submit       : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
  complete     : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
  issued r/w/d: total=0/978992/0, short=0/0/0
  lat (usec): 50=0.02%, 100=98.30%, 250=1.06%, 500=0.01%, 750=0.01%
  lat (usec): 1000=0.01%
  lat (msec): 2=0.01%, 4=0.01%, 10=0.25%, 20=0.29%, 50=0.06%
  lat (msec): 100=0.01%, 250=0.01%
```

fio
Flexible I/O tester (Linux)
<https://fio.readthedocs.io>




Phoronix Test Suite (<https://www.phoronix-test-suite.com>)

- Multiplataforma: Linux, Solaris, Mac OS X, Windows, entre otros S.O.
- Código abierto (<https://github.com/phoronix-test-suite>).
- Permite la instalación, ejecución y la generación de informes tanto de tests de rendimiento (*test profiles*, <https://openbenchmarking.org/tests>) como de agrupaciones de éstos (*test suites*, <https://openbenchmarking.org/suites>) de forma automatizada.
- Permite recopilar automáticamente en tiempo de ejecución la información de sensores del sistema (consumo de energía, temperaturas, voltajes, frecuencias CPU, velocidad ventiladores, utilización de CPU/GPU/discos/memoria, etc.)
- Permite visualizar el resultado de forma gráfica en html y/o exportarlo a <https://openbenchmarking.org> para compartirlo con otros y/o compararlo con otros resultados.



El paquete de microbenchmarks SPEC CPU 2017

- SPEC (*Standard Performance Evaluation Corporation*), es una corporación *sin ánimo de lucro* cuyo propósito es establecer, mantener y respaldar la estandarización de benchmarks y herramientas para evaluar el rendimiento y la eficiencia energética de los equipos informáticos.
- Miembros de la corporación (<https://www.spec.org/consortium/>): *Acer, AMD, Amazon Web Services, Apple, ARM, ASUS, Cisco, Dell, Fujitsu, Google, HP, IBM, Intel, Microsoft, Oracle, Qualcomm, Red Hat, Samsung...*
- Compuesto por cuatro conjuntos de microbenchmarks distintos (<http://www.spec.org/cpu2017/>):
 -  SPEC_{speed}®2017 Integer (rendimiento en aritmética entera)
 - SPEC_{speed}®2017 Floating Point (rendimiento en coma flotante)
 - SPEC_{rate}®2017 Integer (rendimiento en aritmética entera)
 - SPEC_{rate}®2017 Floating Point (rendimiento en coma flotante)
 - Speed: cuánto tarda en ejecutarse un programa (tiempo de respuesta).
 - Rate: cuántos programas puedo ejecutar por unidad de tiempo (productividad).
- ¿Qué componentes se evalúan? Procesador (enteros o coma flotante según el caso), sistema de memoria y compilador (C, Fortran y C++).

El paquete de microbenchmarks SPEC CPU 2017 (cont.)

- SPEC CPU2017 se distribuye como una imagen ISO que contiene:
 - Código fuente de todos los programas de benchmark.
 - Data sets que necesitan algunos benchmarks para su ejecución.
 - Herramientas varias para compilación, ejecución, obtención de resultados, validación y generación de informes.
 - Documentación, incluyendo reglas de ejecución y de generación de informes.
- Reglas para validar los resultados: <https://www.spec.org/cpu2017/Docs/runrules.html>
- El tiempo de ejecución depende del índice a obtener, la máquina en la que se ejecuta y cuántas copias o subprocesos se eligen.

Metric	Config Tested	Individual benchmarks	Full Run (Reportable)
-----	-----	-----	-----
SPECrate2017_int_base	1 copy	6 to 10 minutes	2.5 hours
SPECrate2017_fp_base	1 copy	5 to 36 minutes	4.8 hours
SPECspeed2017_int_base	4 threads	6 to 15 minutes	3.1 hours
SPECspeed2017_fp_base	16 threads	6 to 75 minutes	4.7 hours

Programas dentro de SPECspeed®2017

- Criterios generales:
 - Han de ser aplicaciones reales.
 - Portabilidad a muchas arquitecturas: Intel y AMD x86 & x86-64, Sun SPARC, IBM POWER e IA-64.
- Ejemplo: SPECspeed®2017 Integer: 10 programas (la mayoría en C y C++)
 - 600.perlbench_s Intérprete de Perl
 - 657.xz_s Utilidad de compresión
 - 602.gcc_s Compilador de C
 - 623.xalancbmk_s Conversión XML a HTML
 - ...
- Ejemplo: SPECspeed®2017 Floating Point: 10 programas (la mayoría en Fortran y C)
 - 619.lbm_s Dinámica de fluidos
 - 621.wrf_s Predicción meteorológica
 - 638.imagick_s Procesamiento de imágenes
 - ...

Índices de prestaciones de SPECspeed®2017

- También llamados, de forma genérica, **índices SPEC**:
 - Aritmética entera: CPU2017IntegerSpeed_**peak**, CPU2017IntegerSpeed_**base**.
 - Aritmética en coma flotante: CPU2017FP_Speed_**peak**, CPU2017FP_Speed_**base**.
- Significado de “base” y “peak”:
 - Base: Compilación en modo conservador: todos los programas escritos en el mismo lenguaje usan las mismas opciones de compilación.
 - Peak: Rendimiento pico, permitiendo que cada uno escoja las opciones de compilación óptimas para cada programa.
- Cálculo: Cada programa del benchmark se ejecuta 3 veces y se escoge el resultado intermedio (se descartan los 2 extremos). **El índice SPEC es la media geométrica de las ganancias en velocidad con respecto a una máquina de referencia** (en SPEC CPU2017 una *Sun Fire V490*).
- Ejemplo: Si llamamos t_i al tiempo que tarda la máquina a evaluar en ejecutar el programa de benchmark i -ésimo y t_i^{REF} lo que tardaría la máquina de referencia para ese programa (y hay 10 programas en el benchmark):

$$\text{índice SPEC} = \sqrt[10]{\frac{t_1^{REF}}{t_1} \times \frac{t_2^{REF}}{t_2} \times \dots \times \frac{t_{10}^{REF}}{t_{10}}}$$

Resultados de SPECspeed®2017 Integer



All SPEC CPU2017 Integer Speed Results Published by SPEC

These results have been submitted to SPEC; see [the disclaimer](#) before studying any results.

[Search published CPU2017 results](#)

Last update: 2017-10-19 11:49

CPU2017 Integer Speed (7):

[Search in CPU2017 Integer Speed results](#)

Test Sponsor	System Name	Parallel	Base Threads	Processor			Results	
				Enabled Cores	Enabled Chips	Threads/ Core	Base	Peak
HPE	Integrity Superdome X (384 core, 2.20 GHz, Intel Xeon E7-8890 v4) HTML CSV Text PDF PS Config	No	384	384	16	2	5.31	5.86
HPE	ProLiant DL580 Gen9 (2.20 GHz, Intel Xeon E7-8890 v4) HTML CSV Text PDF PS Config	No	96	96	4	1	5.35	5.95
HPE	ProLiant ML350 Gen9 (2.20 GHz, Intel Xeon E5-2699 v4) HTML CSV Text PDF PS Config	No	44	44	2	1	5.80	6.43
HPE	ProLiant DL380 Gen10 (2.10 GHz, Intel Xeon Platinum 8170) HTML CSV Text PDF PS Config	Yes	52	52	2	1	8.96	Not Run
HPE	ProLiant DL380 Gen10 (2.10 GHz, Intel Xeon Platinum 8176) HTML CSV Text PDF PS Config	Yes	56	56	2	1	9.16	Not Run
Huawei	Huawei 2288H V5 (Intel Xeon Platinum 8180) HTML CSV Text PDF PS Config	Yes	56	56	2	1	9.46	9.79
Oracle Corporation	Sun Fire V490 HTML CSV Text PDF PS Config	Yes	1	8	4	1	1.00	Not Run

Resultados de SPECspeed® 2017 Integer (II)

Hardware		Software	
CPU Name:	Intel Xeon E7-8890 v4	OS:	SUSE Linux Enterprise Server 12 (x86_64) SP1
Max MHz:	3400		3.12.53-60.30-default
Nominal:	2200	Compiler:	C/C++: Version 17.0.0.098 of Intel C/C++
Enabled:	384 cores, 16 chips, 2 threads/core		Compiler for Linux;
Orderable:	2 to 16 chips		Fortran: Version 17.0.0.098 of Intel Fortran
Cache L1:	32 KB I + 32 KB D on chip per core		Compiler for Linux
L2:	256 KB I+D on chip per core	Parallel:	No
L3:	60 MB I+D on chip per chip	Firmware:	HP Bundle: 008.004.084 SFW: 043.025.000 08/16/2016
Other:	None	File System:	xfs
Memory:	4 TB (128 x 32 GB 2Rx4 PC4-2400T-L, running at 1600 MHz)	System State:	Run level 5 (multi-user, w/GUI)
Storage:	8 x C8S59A, 900 GB 10 K RPM SAS	Base Pointers:	64-bit
Other:	None	Peak Pointers:	32/64-bit
		Other:	Microquill SmartHeap V10.2

Results Table

Benchmark	Base							Peak						
	Threads	Seconds	Ratio	Seconds	Ratio	Seconds	Ratio	Threads	Seconds	Ratio	Seconds	Ratio	Seconds	Ratio
600.perlbench_s	384	365	4.86	358	4.96	357	4.98	384	298	5.95	295	6.02	295	6.01
602.gcc_s	384	553	7.20	546	7.29	546	7.29	384	540	7.37	535	7.45	534	7.45
605.mcf_s	384	866	5.45	866	5.45	898	5.26	384	708	6.67	700	6.75	699	6.75
620.omnetpp_s	384	276	5.90	271	6.03	289	5.65	384	251	6.50	247	6.61	246	6.64
623.xalancbmk_s	384	189	7.50	188	7.52	187	7.57	384	179	7.91	179	7.93	180	7.87
625.x264_s	384	283	6.24	282	6.25	283	6.23	384	271	6.51	272	6.49	270	6.52
631.deepsjeng_s	384	407	3.52	408	3.52	407	3.52	384	343	4.18	343	4.18	343	4.18
641.leela_s	384	460	3.64	460	3.63	460	3.63	384	438	3.00	430	3.88	440	3.88

Resultados de SPECspeed®2017 Integer (III)

Base Optimization Flags

C benchmarks:

-qopt-prefetch -qopt-mem-layout-trans=3 -DSPEC_SUPPRESS_OPENMP

C++ benchmarks:

-Wl,-z,muldefs -qopt-prefetch -qopt-mem-layout-trans=3 -DSPEC_SUPPRESS_OPENMP -L/sh10.2 -lsmartheap64

Peak Optimization Flags

C benchmarks:

600.perlbench_s: -prof-gen(pass 1) -prof-use(pass 2) -O2 -xCORE-AVX2 -auto-p32 -ipo -qopt-prefetch -O3 -no-prec-div
-qopt-mem-layout-trans=3 -DSPEC_SUPPRESS_OPENMP

602.gcc_s: Same as 600.perlbench_s

605.mcf_s: -prof-gen(pass 1) -prof-use(pass 2) -ipo -xCORE-AVX2 -O3 -no-prec-div -qopt-prefetch -qopt-mem-layout-trans=3
-DSPEC_SUPPRESS_OPENMP

625.x264_s: Same as 600.perlbench_s

657.xz_s: Same as 600.perlbench_s

C++ benchmarks:

620.omnetpp_s: -Wl,-z,muldefs -prof-gen(pass 1) -prof-use(pass 2) -ipo -xCORE-AVX2 -O3 -no-prec-div -auto-p32 -qopt-prefetch
-qopt-mem-layout-trans=3 -DSPEC_SUPPRESS_OPENMP -L/sh10.2 -lsmartheap64

623.xalancbmk_s: Same as 620.omnetpp_s

631.deepsjeng_s: -Wl,-z,muldefs -prof-gen(pass 1) -prof-use(pass 2) -ipo -xCORE-AVX2 -O3 -no-prec-div -qopt-prefetch
-qopt-mem-layout-trans=3 -DSPEC_SUPPRESS_OPENMP -L/sh10.2 -lsmartheap64

641.leela_s: Same as 620.omnetpp_s

Ejemplo de cálculo de SPECspeed®2017 Integer_{base}

Benchmark	t ^{REF} (s)	Exp1 (s)	Exp2 (s)	Exp3 (s)	t ^{base} (s)	t ^{REF} / t ^{base}
600.perlbench_s	1774	365	358	357	358	4,96
602.gcc_s	3981	553	546	546	546	7,29
605.mcf_s	4721	866	866	898	866	5,45
620.omnetpp_s	1630	276	271	289	276	5,91
623.xalancbmk_s	1417	189	188	187	188	7,54
625.x264_s	1764	283	282	283	283	6,23
631.deepsjeng_s	1432	407	408	407	407	3,52
641.leela_s	1706	469	469	469	469	3,64
648.exchange2_s	2939	329	329	329	329	8,93
657.xz_s	6182	2165	2161	2164	2164	2,86

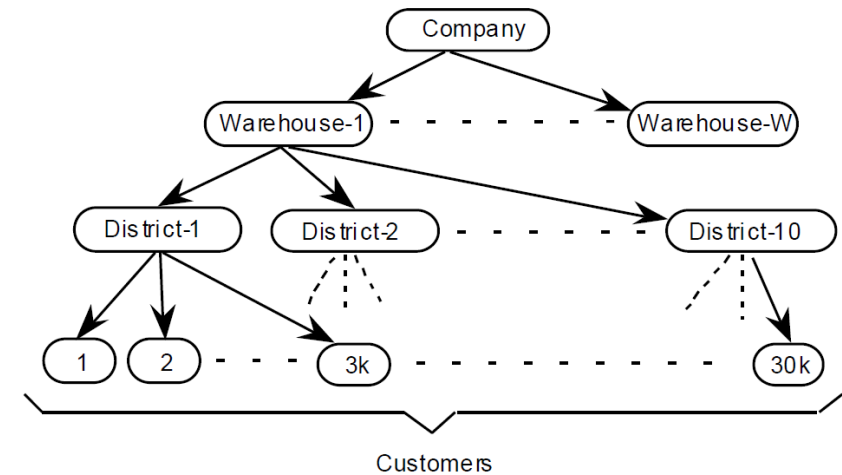
$$\text{SPECspeed®2017 Integer}_{\text{base}} = \sqrt[10]{\frac{t_1^{\text{REF}}}{t_1^{\text{base}}} \times \frac{t_2^{\text{REF}}}{t_2^{\text{base}}} \times \dots \times \frac{t_{10}^{\text{REF}}}{t_{10}^{\text{base}}} = \sqrt[10]{4,96 \times 7,29 \times 5,45 \times \dots} = 5,31$$

Benchmarks de sistema completo: TPC

- TPC (*Transactions Processing Performance Council*, <http://www.tpc.org>): Organización sin ánimo de lucro especializada en benchmarks relacionados con servidores que procesan datos (*data-driven servers*).
- Empresas que hay detrás de TPC (<http://tpc.org/information/who/howeare5.asp>): *AMD, Cisco, Dell, Fujitsu, HP, Huawei, IBM, Intel, Microsoft, Nvidia, Oracle, ...*
- Principales **tipos** de benchmarks:
 - OLTP (*on-line transaction processing*): Se trata de realizar pequeñas transacciones (compras, consultas,...) con unos requisitos exigentes de tiempos de respuesta.
 - DS (*decision support*): Se trata de realizar consultas complejas que suelen requerir acceso a buena parte de una gran base de datos y un procesamiento posterior.
 - IoT (*Internet of Things*): Se trata de procesar una gran cantidad de datos procedentes de una gran cantidad de dispositivos diferentes.
 - Big Data: Se trata de procesar una gran cantidad de información usando clusters de computadores y [Apache Hadoop](#).
 - Artificial Intelligence: Se trata de ejecutar algoritmos de aprendizaje utilizados en *machine learning* (p.ej. redes neuronales profundas) a partir de datos.
 - Virtualization: Se trata de ejecutar los benchmarks de los tipos anteriores en un entorno virtualizado en lugar de en una máquina “física” (*bare metal*).

Ejemplo: Benchmark TPC-C

- Es de tipo OLTP (*on-line transaction processing*). Simula una gran compañía que vende 100.000 productos y que tiene varios almacenes (configurable), cada uno a cargo de 10 zonas, con 3000 clientes/zona. Las peticiones involucran acceso a las bases de datos tanto locales como distribuidas (a veces el producto no está en el almacén más cercano), ejecución simultánea de consultas y acceso no uniforme a las bases de datos.



- Índice de rendimiento utilizado: transacciones procesadas por segundo, minuto u hora (*tps*, *tpm* o *tph*) superando unos ciertos requisitos de tiempos de respuesta (ej. el 90% deben tener un tiempo de respuesta inferior a 5s). También suelen proporcionar tanto el consumo de potencia como el coste por transacción procesada (incluido mantenimiento de 3 años).

Ejemplo: Benchmark TPC-H

- Es de tipo DS (*decision support*). Contiene un total de 22 tipos de consultas diferentes que requieren examinar grandes volúmenes de datos para poder contestar a preguntas complejas. También incluyen modificaciones concurrentes de los datos. Algunos de estos tipos de consultas son:
 - Realizar un informe detallado sobre las ventas en un determinado periodo de tiempo.
 - Buscar el proveedor más adecuado según un conjunto determinado de criterios.
 - Cuantificar el aumento de ingresos que habría resultado de eliminar ciertos descuentos en toda la empresa en un rango de fechas determinado.
- Aunque es escalable, la base de datos cuenta con la información de un mínimo de 10000 proveedores con un mínimo de 10 millones de filas.
- Índice de rendimiento utilizado (métrica): *TPC-H Composite Query-per-Hour Performance Metric (QphH@Size)*, donde *Size* es el tamaño de la base de datos utilizado (factor de escala). Tiene en cuenta la productividad (consultas por unidad de tiempo) en dos escenarios diferentes: cuando las consultas se realizan por un único usuario y cuando se permite concurrencia entre varios usuarios. También suelen proporcionar tanto el consumo de potencia como el coste por *QphH@Size* (incluido mantenimiento de 3 años).

TPC-H: Búsqueda de resultados

TPC™
disseminating objective, verifiable performance data to the industry... The TPC is a non-profit corporation focused on developing data-centric benchmarks

Document Search

Home
About the TPC
Who We Are
Contact Us
Privacy Policy
Stay Connected
Press
TPC FAQ
History
Benchmarks
Enterprise BMs
TPC-C
TPC-DS
TPC-E
TPC-H
TPC-VMS
Express BMs
TPCx-BB
TPCx-HS
TPCx-V
Common Specifications
TPC-Pricing
TPC-Energy
Submission Checklist
Obsolete BMs
- TPC-A
- TPC-App
- TPC-B
- TPC-D
- TPC-R

TPC-H Advanced Sort Results List (V2.2) As of 25-Oct-2017 at 08:51 [Pacific Time Zone]

Note 1: The TPC believes that comparisons of TPC-H results measured against different database sizes are misleading and discourages such comparisons.
Note 2: The TPC believes it is not valid to compare prices or price/performance of results in different currencies.

Filter Options
Scale Factor: ☐ Enter Numeric
[Add Row](#)

Sort Options
Availability Date: [Add Row](#)

Display Options
of results to display:
[Display withdrawn results:](#)
[Display Historical Results:](#)
☐ Specification Revision
☐ Total System Price
☐ OS Software Name
☐ Server CPU Name & Processors/Co
☐ Cluster
☐ Include Energy Data
[Show Results](#)

Color Legend
• Results displayed which

TPC-H Advanced Sorting Results

Sponsor	System	Scale Factor	Performance (QphH)	Price/QphH	System Availability	Date Submitted	DB Software Name
Hewlett Packard Enterprise	HPE ProLiant DL380 Gen9	1,000	717,101	0.61 USD	10/19/2017	4/17/2017	Microsoft SQL Server 2017 Enterprise Edition
Hewlett Packard Enterprise	HPE ProLiant DL380 Gen9	1,000	543,102	0.69 USD	7/31/2016	3/9/2016	Microsoft SQL Server 2016 Enterprise Edition
Lenovo	Lenovo System x3850 X6	3,000	969,504	0.72 USD	7/31/2016	3/9/2016	Microsoft SQL Server 2016 Enterprise Edition
Hewlett Packard Enterprise	HPE ProLiant DL380 Gen9	1,000	678,492	0.64 USD	7/31/2016	3/24/2016	Microsoft SQL Server 2016 Enterprise Edition
Hewlett Packard Enterprise	HPE ProLiant DL580 Gen9 Action Vector 5.0	3,000	2,140,307	0.38 USD	7/31/2016	6/2/2016	Action Vector 5.0
CISCO	Cisco UCS C460 M4 Server	3,000	1,071,018	0.60 USD	6/1/2016	5/14/2016	Microsoft SQL Server 2016 Enterprise Edition
CISCO	Cisco UCS C460 M4 Server	3,000	725,686	1.08 USD	7/14/2015	7/13/2015	Microsoft SQL Server 2014 Enterprise Edition
Lenovo	Lenovo System x3850 X6	3,000	700,392	0.99 USD	5/26/2015	5/1/2015	Microsoft SQL Server 2014 Enterprise Edition

Base de datos de tamaño ≤ 3000 GB

TPC-H: Búsqueda de resultados

TPC-H Result Highlights As of 25-Oct-2017 at 3:54 PM [GMT]

CISCO **Cisco UCS C460 M4 Server**
Reference URL: <http://www.tpc.org/3322>

Benchmark Stats

Result ID:	116051401
Status:	Accepted Result
Report Date:	05/14/16
TPC-H Rev:	2.17.1

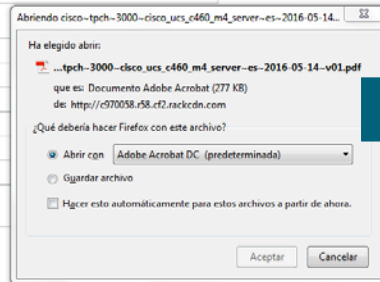
System Information


Total System Cost:	634,322 USD
Performance:	1,071,018 QphH@3000GB
Price/Performance:	.60 USD per QphH@3000GB
TPC Energy Metric:	Not reported
Availability Date:	06/01/16
Database Manager:	Microsoft SQL Server 2016 Enterprise Edition
Operating System:	Microsoft Windows Server 2012 R2 Standard Edition


Server Specific Information

CPU Type:	Intel Xeon E7-8890 v3 2.50GHz
Total # of Processors:	4
Total # of Cores:	72
Total # of Threads:	144
Cluster:	No
Load Time (hours):	1.94
Total Storage/Database Size Ratio:	2.99

☐ Executive Summary
☐ Full Disclosure Report
☐ Supporting Files-1



		Cisco UCS C460 M4 Server		TPC-H Rev. 2.17.1 TPC-Pricing Rev. 2.0.0 Report Date: 16-May-2016	
Total System Cost		Composite Query per Hour Metric		Price / Performance	
\$634,322 USD		1,071,018.2 QphH@3000GB		\$0.60 USD \$/ QphH@3000GB	
Database Size	Database Manager	Operating System	Other Software	Availability Date	
3000GB	Microsoft SQL Server 2016 Enterprise Edition	Windows 2012 R2 Standard Edition		1-June-2016	

			
Database Load Time = 1h 56m 26s		Storage Redundancy Level	
Load Includes Backup: Y		Base Tables and Auxiliary Data Structures	0
Total Data Storage / Database Size = 2.99		DBMS Temporary Space	0
Percentage Memory / Database Size = 102.4%		OS and DBMS Software	1
System Configuration: Cisco UCS C460 M4 Server			
Processors/Cores/Threads/Model:		4/72/144 Intel Xeon E7-8890 v3 Processor (2.5 GHz, 45MB cache, 165W)	
Memory:		3 TB	
Storage:		8 X 400GB 2.5 inch Ent Performance 12G SAS SSD (10X endurance)	
Table Storage:		4 X UCS Rack PCIe Storage 1600 GB SanDisk SX350 Medium Endurance	
		9.38 TB	

TPC-H: Búsqueda de resultados



Server Hardware

Description	Unit Price	Qty	Extended Price
UCS C460 M4 base chassis w/o CPU/DIMM/HDD	16,500.00	1	\$16,500.00
3YR SNTC 24X7X4OS UCS C460 M4 Server	3,487.00	1	
2.5GHz E7-8890 v3/165W/18C/45M Cache	21,000.00	4	\$84,000.00
32GB DDR4-2133-MHz RDIMM/PC4-17000/dual rank/x4/1.2v	1,100.00	96	\$105,600.00
UCS C460 M4 DDR4 Memory Riser with 12 DIMM slots	800.00	8	\$6,400.00
Riser card with 5 PCIe slots	500.00	2	\$1,000.00
400GB 2.5 inch Ent Performance 12G SAS SSD (10X endurance)	5,267.00	8	\$42,136.00
1400W V2 AC Power Supply (200 - 240V) 2U & 4U C Series	800.00	4	\$3,200.00
Power Cord, 200/240V 6A North America	0.00	4	\$0.00
Full Height PCIe slot filler for C Series	0.00	6	\$0.00
Bracket and Supercap cable for C460 M4 and 12 drive RAID	0.00	1	\$0.00
CPU Heat Sink for UCS C460 M4 Rack Server	0.00	4	\$0.00
Rail Kit for UCS C460 M4	0.00	1	\$0.00
Cisco 12G SAS Modular Raid Controller (12 port)	1,688.00	1	\$1,688.00
Cisco 12Gbps SAS 1GB FBWC Cache module (Raid 0/1/5/6)	1,217.00	1	\$1,217.00
Cisco ONE Data Center Compute Opt Out Option	0.00	1	\$0.00
UCS 2.5 inch HDD blanking panel	0.00	4	\$0.00
UCS Rack PCIe Storage 1600GB SanDisk SX350 Medium Endurance	18,133.00	4	\$72,532.00
Cisco R42610 standard rack, w/side panels	3,429.00	1	\$3,429.00
IOGEARGKM513 Spill Proof Keyboard & Mouse Combo	15.91	1	\$15.91
ASUS 19.5" VS207D-P Widescreen LED 1600x900 VGA	87.71	1	\$87.71
			<u>\$337,806</u>

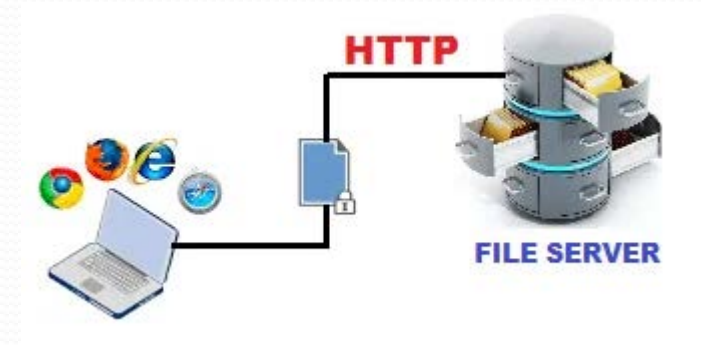


Podemos usar el benchmarking para tener una idea de cómo diseñar nuestro servidor.



Otros benchmarks de sistema completo

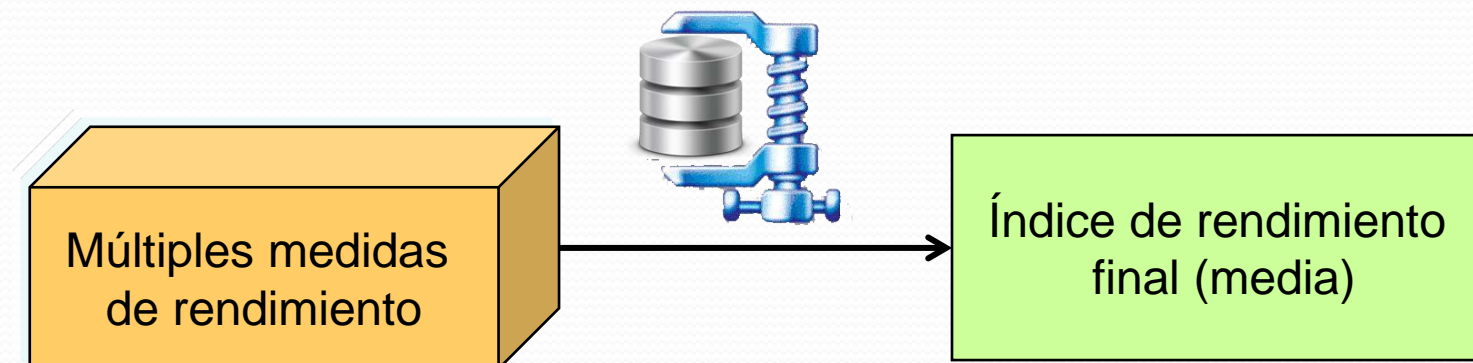
- **Servidores de ficheros:**
 - SPECstorage Solution 2020.
 - SPEC SFS2014.
- **JAVA Cliente/Servidor:**
 - SPECjEnterprise2010: Java Enterprise Edition (JEE).
 - SPECjms2007: Java Message Service (JMS).
 - SPECjvm2008: Java Runtime Environment (JRE).
- **High Performance Computing, OpenMP, MPI, OpenCL:**
 - SPEC MPI2007: Message Passing Interface (MPI).
 - SPEC OMP2012: Open MultiProcessing (OpenMP).
 - SPEC ACCEL: OpenCL y OpenACC.
- **Para el mundo del PC:**
 - SPECworkstation® 3.1: Sistemas basados en Windows. Usa programas de código abierto.
 - SYSmark25: De la empresa BAPco. Sistemas basados en Windows. Usa programas propietarios.



4.2. Análisis de los resultados de un test de rendimiento

¿Cómo expresar el resultado final tras la ejecución de un test de rendimiento?

- Muchos tests de rendimiento se basan en la ejecución de diferentes programas y, por tanto, producen diferentes medidas de rendimiento.
- Sin embargo, estos tests suelen resumir todas estas medidas en un **único** valor: el índice de rendimiento de dicho test.
- ¿Cómo concentrar todos los índices en uno solo?
 - Método habitual de síntesis: uso de algún tipo de **media**.



La media aritmética

- Dado un conjunto de n medidas, t_1, \dots, t_n , definimos su media aritmética:

$$\bar{t} = \frac{1}{n} \sum_{k=1}^n t_k$$

- Si no todas las medidas tienen la misma importancia, se puede asociar a cada medida t_k un peso w_k , obteniéndose la **media aritmética ponderada**:

$$\overline{t_W} = \sum_{k=1}^n w_k \times t_k$$

$$\text{con } \sum_{k=1}^n w_k = 1$$

Si t_k es el tiempo de ejecución del programa de benchmark k -ésimo en la máquina a testar, w_k podría escogerse, por ejemplo, inversamente proporcional a dicho tiempo de ejecución en una determinada máquina de referencia:

$$w_k \equiv \frac{C}{t_k^{REF}}$$



$$C = \frac{1}{\sum_{k=1}^n 1/t_k^{REF}}$$

La media geométrica

- Dado un conjunto de n medidas, S_1, \dots, S_n , definimos su media geométrica:

$$\overline{S_g} = \sqrt[n]{\prod_{k=1}^n S_k} = \left(\prod_{k=1}^n S_k \right)^{1/n}$$

- Propiedad: cuando las medidas son ganancias en velocidad (*speedups*) con respecto a una máquina de referencia, este índice mantiene el mismo orden en las comparaciones independientemente de la máquina de referencia elegida (siempre que sea la misma). Usado en los benchmarks de SPEC y SYSmark25.

$$SPEC(M) = \sqrt[n]{\frac{t_1^{REF}}{t_1^M} \times \frac{t_2^{REF}}{t_2^M} \times \dots \times \frac{t_n^{REF}}{t_n^M}} = \frac{\sqrt[n]{t_1^{REF} \times t_2^{REF} \times \dots \times t_n^{REF}}}{\sqrt[n]{t_1^M \times t_2^M \times \dots \times t_n^M}}$$

$$SPEC(M1) > SPEC(M2) \Leftrightarrow \sqrt[n]{t_1^{M1} \times t_2^{M1} \times \dots \times t_n^{M1}} < \sqrt[n]{t_1^{M2} \times t_2^{M2} \times \dots \times t_n^{M2}}$$

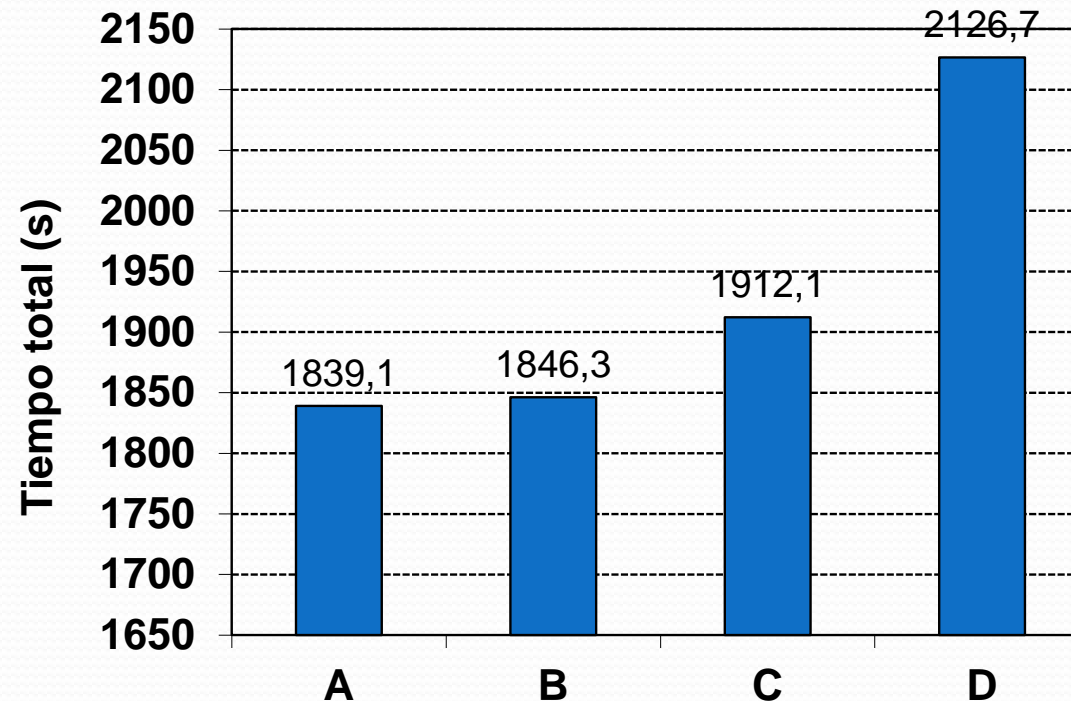
Ejemplo de comparación con tiempos

Programa	t^{REF} (s)	t^A (s)	t^B (s)	t^C (s)	t^D (s)
1	1400	141	170	136	134
2	1400	154	166	215	25
3	1100	96,8	94,2	146	201
4	1800	271	283	428	523
5	1000	83,8	90,1	77,4	81,2
6	1200	179	189	199	245
7	1300	120	131	87,7	75,5
8	300	151	158	138	192
9	1100	93,5	122	88	118
10	1900	133	173	118	142
11	1500	173	170	179	240
12	3000	243	100	100	150
Suma	17000	1839,1	1846,3	1912,1	2126,7

- La máquina más rápida es “A” ya que es la que tarda menos en ejecutar, uno tras otro, todos los programas del benchmark (1839,1 segundos).

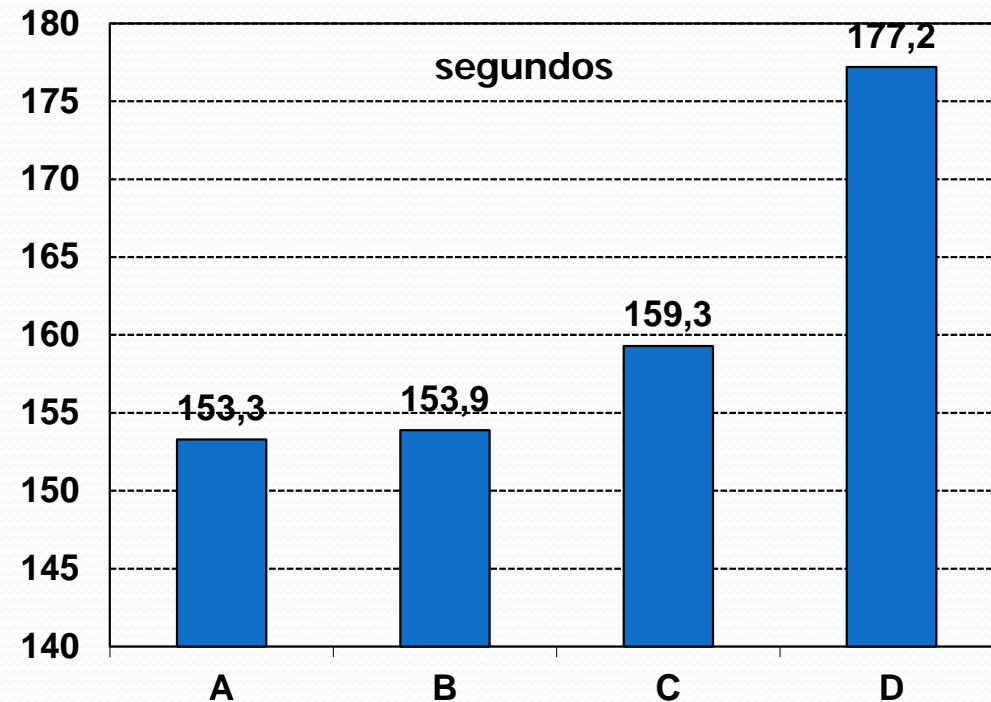
Comparación con el tiempo total

- Ordenación con el tiempo total:
 - De más rápida a más lenta: A, B, C, D
 - Esto no significa que A sea siempre la más rápida (depende del programa), aunque, en conjunto, sí que lo es.



Comparación con la media aritmética

$$\begin{aligned}\bar{t}_A &= \frac{1}{12} \sum_{k=1}^{12} t_k^A = 153,3s \\ \bar{t}_B &= \frac{1}{12} \sum_{k=1}^{12} t_k^B = 153,9s \\ \bar{t}_C &= \frac{1}{12} \sum_{k=1}^{12} t_k^C = 159,3s \\ \bar{t}_D &= \frac{1}{12} \sum_{k=1}^{12} t_k^D = 177,2s\end{aligned}$$



- La máquina que ejecuta los programas del benchmark, uno tras otro, en menor tiempo es la de menor media aritmética de los tiempos de ejecución.

Usando la media aritmética ponderada

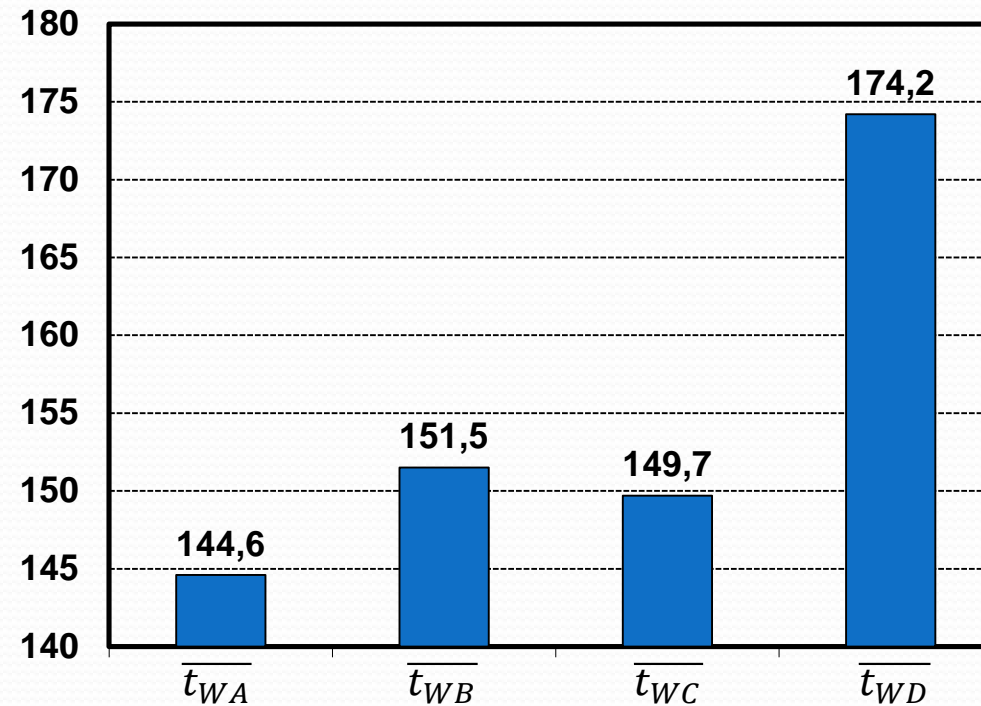
Prog	t_k^{REF} (s)	w_k
1	1400	0,06
2	1400	0,06
3	1100	0,08
4	1800	0,05
5	1000	0,09
6	1200	0,07
7	1300	0,07
8	300	0,30
9	1100	0,08
10	1900	0,05
11	1500	0,06
12	3000	0,03
Suma	17000	1

$$w_k \equiv \frac{C}{t_k^{REF}}$$

$$C = \frac{1}{\sum_{k=1}^n 1/t_k^{REF}} = 88,77s$$

$$\overline{t_{WA}} = \sum_{k=1}^{12} w_k \times t_k^A = 144,6s$$

Igualmente, se calculan:
 $\overline{t_{WB}}$, $\overline{t_{WC}}$ y $\overline{t_{WD}}$



- Según este criterio, la máquina “más rápida” sería la de menor tiempo medio ponderado de ejecución. Nótese que esta ponderación depende, en este ejemplo, de la máquina de referencia.

Usando la media geométrica de *speedups*

- Calculamos la ganancia en velocidad de cada máquina con respecto a la máquina de referencia (tal y como lo hacen SPEC y SYSmark25):

Programa	$t^{\text{REF}}(s)$	s^A speedup	s^B speedup	s^C speedup	s^D speedup
1	1400	9,9	8,2	10,3	10,4
2	1400	9,1	8,4	6,5	56,0
3	1100	11,4	11,7	7,5	5,5
4	1800	6,6	6,4	4,2	3,4
5	1000	11,9	11,1	12,9	12,3
6	1200	6,7	6,3	6,0	4,9
7	1300	10,8	9,9	14,8	17,2
8	300	2,0	1,9	2,2	1,6
9	1100	11,8	9,0	12,5	9,3
10	1900	14,3	11,0	16,1	13,4
11	1500	8,7	8,8	8,4	6,3
12	3000	12,3	30,0	30,0	20,0
M. Geom.		8,78	8,66	8,97	9,00

- El *speedup* es un índice a maximizar. Según este índice, la “mejor máquina” es ¡¡¡la D!!!

¿A quién beneficia la decisión de usar la media geométrica de *speedups*?

J8						=MEDIA.GEOM(J2:J5)				
	A	B	C	D	E	F	G	H	I	J
	Prog. Bench.	tREF(s)	tA(s)	tB(s)	tC(s)	tD(s)	tREF/tA	tREF/tB	tREF/tC	tREF/tD
1										
2	1	200	100	99	1	1	2,00	2,02	200,0	200,0
3	2	200	100	101	133	1	2,00	1,98	1,50	200,0
4	3	200	100	100	133	1	2,00	2,00	1,50	200,0
5	4	200	100	100	133	397	2,00	2,00	1,50	0,50
6	Suma	800	400	400	400	400				
7										
8				Media Geométrica			2,0000	2,0001	5,11	44,81



Se premian las mejoras sustanciales. No se castigan empeoramientos no tan sustanciales. Debemos ser MUY cuidadosos con las comparaciones y saber qué estamos haciendo realmente.



Conclusiones de este análisis

- Intentar reducir un conjunto de medidas de un test de rendimiento a un solo “valor medio” final no es una tarea trivial.
- La media aritmética de los tiempos de ejecución es una medida fácilmente interpretable e independiente de ninguna máquina de referencia. El menor valor nos indica la máquina que ha ejecutado el **conjunto** de programas del test, uno tras otro, en un tiempo menor.
- La media aritmética ponderada nos permite asignar más peso a algunos programas que a otros. Esa ponderación debería realizarse, idealmente, según las necesidades del usuario. Si se hace de forma dependiente de los tiempos de ejecución de una máquina de referencia, la elección de ésta puede influir significativamente en los resultados.
- La media geométrica de las ganancias en velocidad con respecto a una máquina de referencia es un índice de interpretación compleja cuya comparación no depende de la máquina de referencia. Premia mejoras sustanciales con respecto a algún programa del test y no castiga al mismo nivel los empeoramientos.

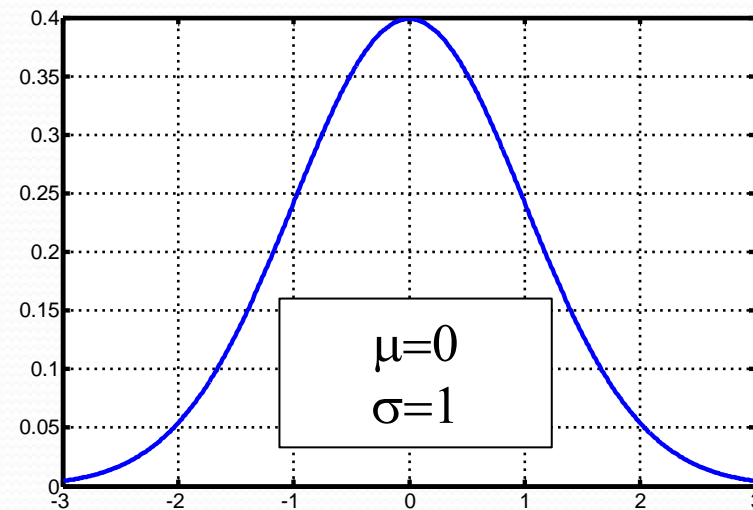
4.3. Comparación de prestaciones en presencia de aleatoriedad

Repaso de Estadística: Distribución Normal

- Independientemente de qué índice se escoja, un buen ingeniero debería, en primer lugar, determinar si las diferencias entre las medidas obtenidas por un test de rendimiento en presencia de aleatoriedad son **estadísticamente significativas** → Necesitaremos repasar algunos conceptos de estadística.
- **Distribución normal:** Es una distribución de probabilidad caracterizada por su media μ y su varianza σ^2 cuya función de probabilidad viene dada por:

$$Prob(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

La probabilidad de obtener un elemento en el rango $[\mu - 2\sigma, \mu + 2\sigma]$ es del 95%



- Teorema del límite central: la media de un conjunto grande de muestras aleatorias de cualquier distribución e independientes entre sí pertenece una distribución normal.

Repaso de Estadística: Distribución t de Student

Si extraemos n muestras $\{d_1, d_2, \dots, d_n\}$ pertenecientes a una distribución Normal de media $\mu = \bar{d}_{real}$, y calculo la siguiente medida (=estadístico):

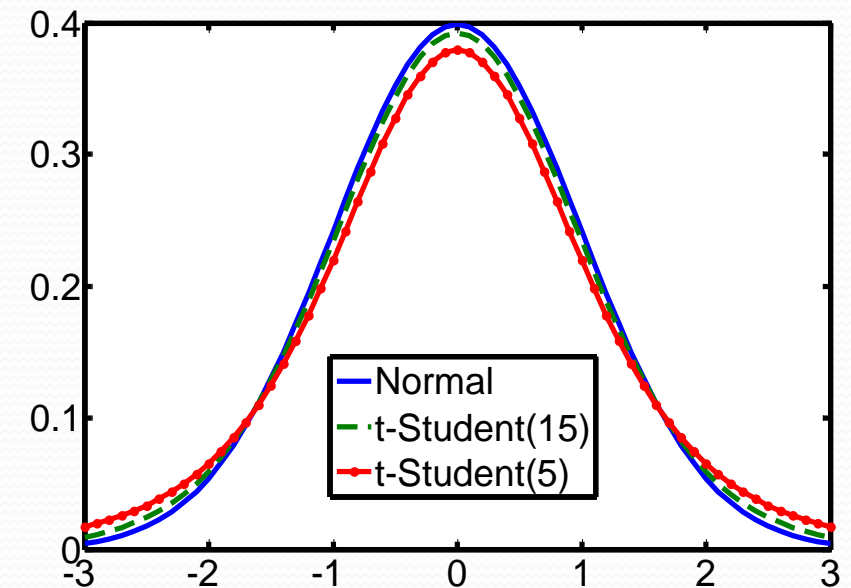
$$t_{exp} = \frac{\bar{d} - \bar{d}_{real}}{s/\sqrt{n}}$$

siendo \bar{d} la media muestral y s la desviación típica muestral

$$\bar{d} = \frac{\sum_{i=1}^n d_i}{n} \quad s = \sqrt{\frac{\sum_{i=1}^n (d_i - \bar{d})^2}{n-1}}$$

y repetimos el experimento muchas veces, veremos que esos t_{exp} pertenecen a una distribución t-Student con $n-1$ grados de libertad (*degrees of freedom, df*).

¿Para qué me puede servir esto?



$$Prob(t) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi} \Gamma(\frac{\nu}{2})} \left(1 + \frac{t^2}{\nu}\right)^{-\frac{\nu+1}{2}}$$

$s/\sqrt{n} \equiv$ Error estándar

Ejemplo 1: Comparación de rendimiento entre A y B

- Tiempos de ejecución (en segundos) de 6 programas (P1...P6) en dos máquinas diferentes (A y B) en condiciones donde puede haber alta aleatoriedad.

Programa	t _A (s)	t _B (s)	d = t _A -t _B (s)
P1	142	100	42
P2	139	92	47
P3	152	128	24
P4	112	82	30
P5	156	148	8
P6	166	171	-5

$$\bar{t}_A = 144,5s$$

$$\bar{t}_B = 120,2s$$

¿Es significativa esta diferencia?

$$\bar{d} = 24,3 s \quad s = 19,9 s \quad s/\sqrt{n} = 8,12 s$$

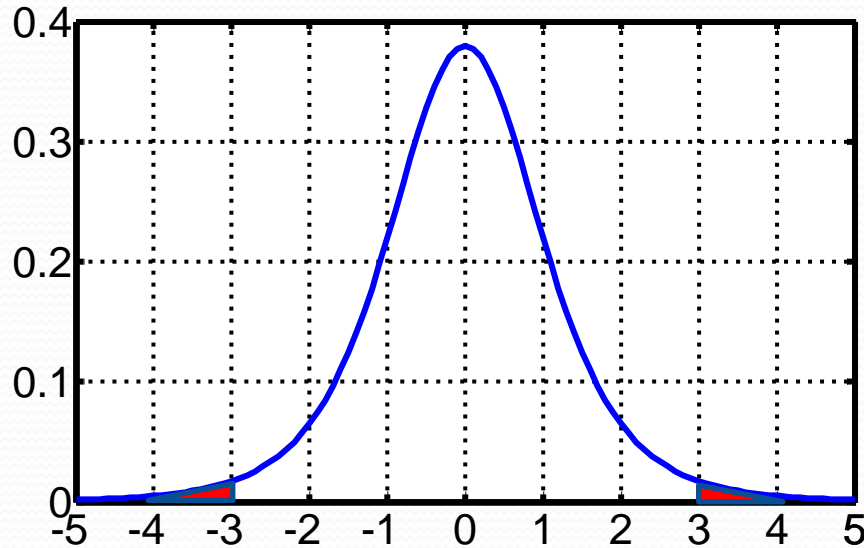
- Si partimos de la hipótesis (hipótesis “nula”, H_0) de que las máquinas tienen rendimientos **equivalentes**, entonces las diferencias se deben a una suma (=una media) de factores aleatorios independientes. En ese caso d_i serán muestras de una distribución normal de media cero ($\bar{d}_{real} = 0$). Por tanto:

$$t_{exp} = \frac{\bar{d}}{s/\sqrt{n}} = \frac{24,3s}{8,12s} = 2,99$$

pertenecerá a una distribución t de Student con 6-1=5 grados de libertad. **¿Qué probabilidad hay de que esto sea realmente así?**

Nivel o Grado de Significatividad (α)

- Distribución t de Student con 5 grados de libertad (T_5).



$$P - \text{value} = P(|t| \geq |t_{exp}|) \text{ en } T_{n-1} \\ = 2 \times P(t \leq -|t_{exp}|) \text{ en } T_{n-1}$$

$$= \text{DISTR.T2C}(2,99;5) = 0,03 \text{ (Excel).}$$

$$= \text{DISTR.T}(2,99;5;2) = 0,03 \text{ (Calc).}$$

$$= 2 \cdot \text{tcdf}(-2,99,5) = 0,03 \text{ (Matlab).}$$

La probabilidad de obtener un valor de $|t|$ igual o superior a 2,99 de una distribución t de Student con 5 grados de libertad es de 0,03 (**P-value** (Valor-P)= 0,03). ¿Es eso mucho o poco? Debemos definir un umbral: **nivel o grado de significatividad α** . Normalmente, $\alpha=0,05$ (5%).

Conclusión: Si $P\text{-value} < \alpha$ diremos que, para un grado de significatividad α o para un **nivel de confianza** $(1-\alpha)*100$ (normalmente 95%), las máquinas tienen rendimientos estadísticamente diferentes. En ese caso, B sería, de media, 1,2 veces más rápida que A en ejecutar cada programa ($144,5/120,2 = 1,2$). En caso contrario, no podríamos descartar la hipótesis de que las máquinas tengan rendimientos equivalentes.

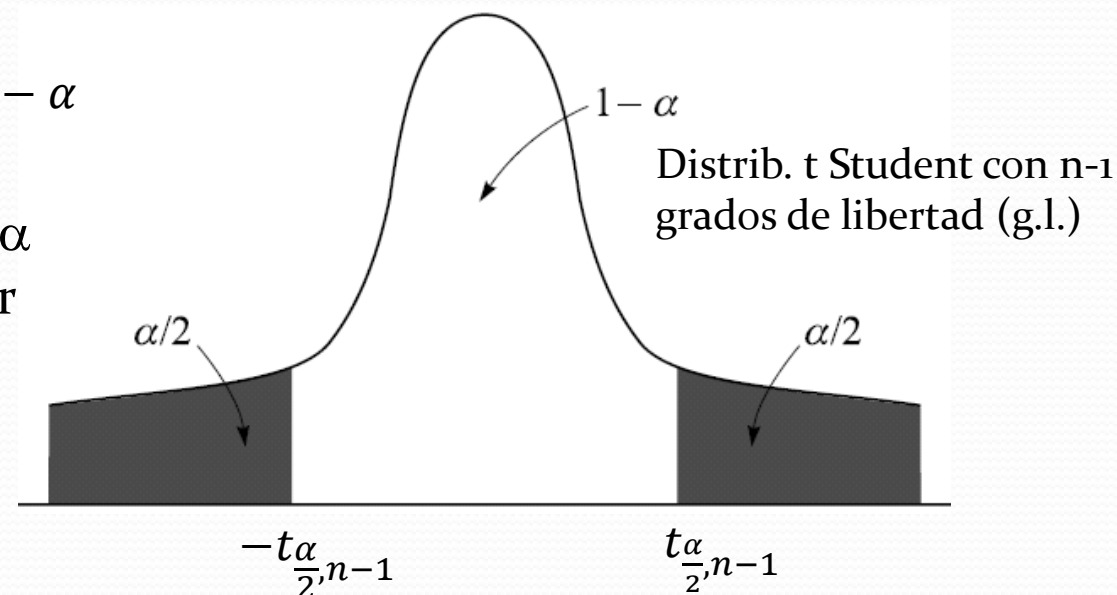
Intervalos de confianza para t_{exp}

- Para un nivel de significatividad α (típ. $0,05 = 5\%$), buscamos el valor $t_{\alpha/2, n-1}$ que cumpla $Prob(|t| > t_{\alpha/2, n-1}) = \alpha$ o equivalentemente:

$$Prob(-t_{\alpha/2, n-1} \leq t \leq t_{\alpha/2, n-1}) = 1 - \alpha$$

- Diremos que para un nivel de confianza $1-\alpha$ (típ. $0,95 = 95\%$), **para aceptar H_0** el valor de t_{exp} debería situarse en el intervalo:

$$[-t_{\alpha/2, n-1}, t_{\alpha/2, n-1}]$$



- A dicho intervalo se le denomina **intervalo de confianza** de la medida para un nivel de significatividad α . Teniendo en cuenta que:

$$Prob(-t_{\alpha/2, n-1} \leq t \leq t_{\alpha/2, n-1}) = 1 - 2 \times Prob(t \leq -t_{\alpha/2, n-1}) = 1 - 2 \times Prob(t > t_{\alpha/2, n-1})$$

es fácil demostrar que $t_{\alpha/2, n-1}$ cumple que (ver figura):

$$Prob(t \leq -t_{\alpha/2, n-1}) = Prob(t > t_{\alpha/2, n-1}) = \alpha/2$$

Intervalos de confianza para t_{exp} (cont.)

- En el caso del *Ejemplo 1*, para un nivel de significatividad de $\alpha=0,05$, buscamos $t_{\alpha/2, n-1}$ tal que:

$$Prob(t \leq -t_{\alpha/2, n-1}) = \alpha/2 = 0,025$$

para una distribución t de Student con 5 grados de libertad. Eso se puede obtener, por ejemplo:

$t_{\alpha/2, n-1}$

- Consultando tablas estadísticas. P.ej. en este [enlace](#) (2-Tail Alpha = 0,05, df = n-1 = 5).
- En *Excel*, haciendo: $ABS(INV.T(alfa/2; n-1)) = ABS(INV.T(0,025; 5)) = 2,57$.
- En *Calc*, $DISTR.T.INV(alfa; n-1) = DISTR.T.INV(0,05; 5) = 2,57$.
- En *Matlab*, haciendo: $abs(tinv(alfa/2, n-1)) = abs(tinv(0,025, 5)) = 2,57$.

- Dicho de otra manera, si las diferencias entre los tiempos de ejecución de ambas máquinas se debieran a factores aleatorios, existiría un 95% de probabilidad de que

$$t_{exp} = \frac{\bar{d}}{s/\sqrt{n}}$$

se encuentre en el rango $[-t_{\alpha/2, n-1}, t_{\alpha/2, n-1}] = [-t_{0,025, 5}, t_{0,025, 5}] = [-2,57, 2,57]$.

Como $t_{exp} = 2,99$ **no** está en ese rango, concluiremos nuevamente que **rechazamos la hipótesis de que ambas máquinas tienen rendimientos equivalentes con el 95% de confianza.**



Intervalos de confianza para \bar{d}_{real}

- Acabamos de ver que si las diferencias entre los tiempos de ejecución de ambas máquinas se debieran a factores aleatorios, existiría un 95% de probabilidad de que t_{exp} se encuentre en el rango $[-t_{\frac{\alpha}{2}, n-1}, t_{\frac{\alpha}{2}, n-1}] = [-2,57, 2,57]$.

- Como

$$t_{exp} = \frac{\bar{d} - \bar{d}_{real}}{s/\sqrt{n}} \in [-t_{\frac{\alpha}{2}, n-1}, t_{\frac{\alpha}{2}, n-1}] = [-2,57, 2,57]$$

sin más que identificar t_{exp} con los valores límite $\pm t_{\frac{\alpha}{2}, n-1}$ sabemos que, de ser H_0 cierta, habrá un 95% de probabilidad de que el valor medio real \bar{d}_{real} de las diferencias entre los tiempos de ejecución se encuentre en el intervalo:

$$\bar{d}_{real} \in \left[\bar{d} - \frac{s}{\sqrt{n}} \times t_{\frac{\alpha}{2}, n-1}, \bar{d} + \frac{s}{\sqrt{n}} \times t_{\frac{\alpha}{2}, n-1} \right] = 24,3 \mp 20,9 = [3,4, 45,2] \text{ s}$$

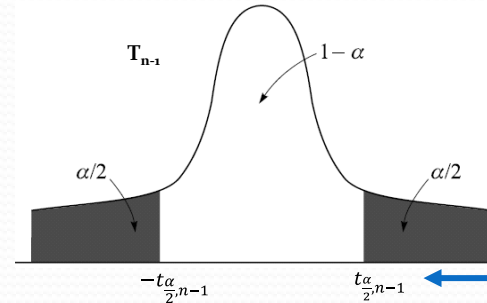
Y el problema se transforma simplemente en comprobar si ese valor medio real \bar{d}_{real} puede o no ser **cero**.

 En nuestro ejemplo, como el intervalo **no** incluye el cero, concluiremos una vez más que **la hipótesis de que ambas máquinas pueden tener rendimientos equivalentes no es cierta al 95% de confianza**.

Resumen: Test t para muestras pareadas

• Partimos de:

Exp.	tA	tB	$d_i = tA_i - tB_i$
P ₁	tA ₁	tB ₁	d ₁
P ₂	tA ₂	tB ₂	d ₂
...
P _n	tA _n	tB _n	d _n



$$df = n-1$$

	α					
df	0.20	0.10	0.05	0.02	0.01	0.001
1	3.0777	6.3138	12.7062	31.8205	63.6567	636.6192
2	1.8856	2.9200	4.3027	6.9646	9.9248	31.5991
3	1.6377	2.3534	3.1824	4.5407	5.8409	12.9240
4	1.5332	2.1318	2.7764	3.7469	4.6041	8.6103
5	1.4759	2.0150	2.5706	3.3649	4.0321	6.8688
6	1.4398	1.9432	2.4469	3.1427	3.7074	5.9588
7	1.4149	1.8946	2.3646	2.9980	3.4995	5.4079
8	1.3968	1.8595	2.3060	2.8965	3.3554	5.0413
9	1.3830	1.8331	2.2622	2.8214	3.2498	4.7809
10	1.3722	1.8125	2.2281	2.7638	3.1693	4.5869
11	1.3634	1.7959	2.2010	2.7181	3.1058	4.4370
12	1.3562	1.7823	2.1788	2.6810	3.0545	4.3178
13	1.3502	1.7709	2.1604	2.6503	3.0123	4.2208
14	1.3450	1.7613	2.1448	2.6245	2.9768	4.1405
15	1.3406	1.7531	2.1314	2.6025	2.9467	4.0728
16	1.3368	1.7459	2.1199	2.5835	2.9208	4.0150
17	1.3334	1.7396	2.1098	2.5669	2.8982	3.9651
18	1.3304	1.7341	2.1009	2.5524	2.8784	3.9216
19	1.3277	1.7291	2.0930	2.5395	2.8609	3.8834
20	1.3253	1.7247	2.0860	2.5280	2.8453	3.8495
21	1.3232	1.7207	2.0796	2.5176	2.8314	3.8193
22	1.3212	1.7171	2.0739	2.5083	2.8188	3.7921
23	1.3195	1.7139	2.0687	2.4999	2.8073	3.7676
24	1.3178	1.7109	2.0639	2.4922	2.7969	3.7454
25	1.3163	1.7081	2.0595	2.4851	2.7874	3.7251
26	1.3150	1.7056	2.0555	2.4786	2.7787	3.7066
27	1.3137	1.7033	2.0518	2.4727	2.7707	3.6896
28	1.3125	1.7011	2.0484	2.4671	2.7633	3.6739
29	1.3114	1.6991	2.0452	2.4620	2.7564	3.6594
30	1.3104	1.6973	2.0423	2.4573	2.7500	3.6460

- Ho: Rendimiento A \equiv Rendimiento B, es decir, $d_i \sim \mathcal{N}(\bar{d}_{real}, \sigma^2)$ con $\bar{d}_{real} = 0$
- Cálculo $t_{exp} = \frac{\bar{d} - \bar{d}_{real}}{s/\sqrt{n}} \sim T_{n-1}$ siendo $\bar{d} = \frac{\sum_{i=1}^n d_i}{n}$ $s = \sqrt{\frac{\sum_{i=1}^n (d_i - \bar{d})^2}{n-1}}$
- Definimos el nivel o grado de significatividad α .
- Rechazamos Ho para un nivel de confianza $(1 - \alpha) \cdot 100(\%)$ si:
 1. Método 1: p-value $< \alpha$. Siendo p-value $= P(|t| \geq |t_{exp}|)$ en $T_{n-1} \approx \text{Prob}(\text{Ho podría ser cierta})$.
 2. Método 2: $t_{exp} \notin [-t_{\alpha/2, n-1}, t_{\alpha/2, n-1}]$. Siendo $t_{\alpha/2, n-1}$ el valor que hace que $\text{Prob}(|t| > t_{\alpha/2, n-1}) = \alpha$ para una distribución t de Student con n-1 grados de libertad.
 3. Método 3: $0 \notin \left[\bar{d} - \frac{s}{\sqrt{n}} \times t_{\alpha/2, n-1}, \bar{d} + \frac{s}{\sqrt{n}} \times t_{\alpha/2, n-1} \right]$. Intervalo de confianza para \bar{d}_{real} .

Ejemplo 2: ¿Influye el parámetro *proxy_cache_min_uses* en este servidor?

- Productividades (en páginas web/s) obtenidas por el servidor en 5 experimentos diferentes para dos valores diferentes (A y B) del parámetro *proxy_cache_min_uses* de Nginx en condiciones donde puede haber alta aleatoriedad.

Experimento	X_A (pág/s)	X_B (pág/s)	$d = X_A - X_B$ (pág/s)
Exp1	23	15	8
Exp2	28	22	6
Exp3	19	20	-1
Exp4	29	27	2
Exp5	36	39	-3

- Usando como criterio la media aritmética ($\overline{X}_A=27$ pág/s, $\overline{X}_B=25$ pág/s) parece que el parámetro A obtiene mejor productividad que el B pero, ¿son significativas las diferencias para un nivel de confianza del 95%? $(1 - \alpha) \times 100 = 95 \rightarrow$ **grado de significatividad $\alpha = 0,05$.**

Ejemplo 2: Realizo el test t para muestras pareadas

- Hago la siguiente hipótesis (H_0):

Rendimiento A \equiv Rendimiento B, es decir, $d_i \sim \mathcal{N}(\bar{d}_{real}, \sigma^2)$ con $\bar{d}_{real} = 0$

- Calculo $t_{exp} = \frac{\bar{d} - \bar{d}_{real}}{s/\sqrt{n}} \sim T_{n-1}$

Experimento	X_A (pág/s)	X_B (pág/s)	$d = X_A - X_B$ (pág/s)
Exp1	23	15	8
Exp2	28	22	6
Exp3	19	20	-1
Exp4	29	27	2
Exp5	36	39	-3

$$\bar{d} = \frac{\sum_{i=1}^n d_i}{n} = \frac{8 + 6 - 1 + 2 - 3}{5} = 2,4 \text{ pág/s}$$

$$s = \sqrt{\frac{\sum_{i=1}^n (d_i - \bar{d})^2}{n-1}} = \sqrt{\frac{(8-2,4)^2 + \dots + (-3-2,4)^2}{5-1}} = 4,6 \text{ pág/s}$$

$$\frac{s}{\sqrt{n}} = \frac{4,6}{\sqrt{5}} = 2,06 \text{ pág/s} \quad t_{exp} = \frac{\bar{d}}{s/\sqrt{n}} = \frac{2,4}{2,06} = 1,16$$

- Método 1:

- p-value = $P(|t| \geq |t_{exp}|)$ en $T_{n-1} = P(|t| \geq |1,16|)$ en $T_4 \approx \text{Prob} (H_0 \text{ podría ser cierta})$.
- Uso *Calc* (por ejemplo): p-value = $\text{DISTR.T}(1,16;4;2) = 0,31$.
- Como p-value $> \alpha$ ($0,31 > 0,05$) no podemos rechazar la hipótesis H_0 al 95% de nivel de confianza (los parámetros A y B sí podrían tener rendimientos equivalentes).

Ejemplo 2: Otros métodos para hacer el test

- Método 2 (intervalos de confianza para t_{exp}):

Calculo $t_{\frac{\alpha}{2}, n-1}$, que es el valor que hace que $Prob(|t| > t_{\frac{\alpha}{2}, n-1}) = \alpha$ para una distribución t de Student con n-1 grados de libertad.

En nuestro caso: $\alpha = 0,05$, $df = n-1 = 4$:

$$Prob(|t| > t_{0,025, 4}) = 0,05$$

Mirando la tabla: $t_{0,025, 4} = 2,78$

α

$df = n-1$	df	0.20	0.10	0.05	0.02	0.01	0.001
1	1	3.0777	6.3138	12.7062	31.8205	63.6567	636.6192
2	2	1.8856	2.9200	4.3027	6.9646	9.9248	31.5991
3	3	1.6377	2.3534	3.1824	4.5407	5.8409	12.9240
4	4	1.5332	2.1318	2.7764	3.7469	4.6041	8.6103
5	5	1.4759	2.0150	2.5706	3.3649	4.0321	6.8688

- Como $t_{\text{exp}} = 1,16 \in [-2,78, 2,78]$ no podemos rechazar la hipótesis H_0 al 95% de nivel de confianza (los parámetros A y B sí podrían tener rendimientos equivalentes).

- Método 3 (intervalos de confianza para \bar{d}_{real}):

$$\left[\bar{d} - \frac{s}{\sqrt{n}} \times t_{\frac{\alpha}{2}, n-1}, \bar{d} + \frac{s}{\sqrt{n}} \times t_{\frac{\alpha}{2}, n-1} \right] = [2,4 - 2,06 \times 2,78, 2,4 + 2,06 \times 2,78] = [-3,3, 8,1] \text{ pág/s.}$$

- Como $0 \in [-3,3, 8,1]$ no podemos rechazar la hipótesis H_0 al 95% de nivel de confianza (los parámetros A y B sí podrían tener rendimientos equivalentes).

Test t con JASP

jasp-stats.org

pairedtests (D:\misd\docencia\milSE\Slides\TEST_T)

Descriptives T-Tests ANOVA

	EJ1_A	EJ1_B	EJ2_A	EJ2_B
1	142	100	23	15
2	139	92	28	22
3	152	128	19	20
4	112	82	29	27
5	156	148	36	39
6	166	171		

pairedtests (D:\misd\docencia\milSE\Slides\TEST_T)

Descriptives T-Tests ANOVA Mixed Models

Classical

- Independent Samples T-Test
- Paired Samples T-Test
- One Sample T-Test

Paired Samples T-Test

EJ1_A
EJ1_B
EJ2_A
EJ2_B

Variable pairs

EJ1_A EJ1_B
EJ2_A EJ2_B

Tests

☒ Student
☐ Wilcoxon signed-rank

Additional Statistics

☒ Location parameter
☒ Confidence interval 95.0 %

degrees of freedom (n-1)

Paired Samples T-Test		t_{exp}	$p - value$		\bar{d}	Standard Error $\frac{s}{\sqrt{n}}$	95% CI for Mean Difference	
Measure 1	Measure 2	t	df	p	Mean Difference	SE Difference	Lower	Upper
EJ1_A	- EJ1_B	2.991	5	0.030	24.333	8.135	3.422	45.245
EJ2_A	- EJ2_B	1.163	4	0.310	2.400	2.064	-3.331	8.131

Intervalo de confianza (95%) para \bar{d}_{real} : $\bar{d} \pm \frac{s}{\sqrt{n}} \times t_{\alpha/2, n-1}$

Otra utilidad del test t: Estimación de intervalos de confianza de medias de medidas experimentales

Hipótesis: Realizamos n medidas $\{d_1, d_2, \dots, d_n\}$ de un mismo fenómeno (p.ej. tiempos de ejecución de un programa, tiempos acceso de un disco duro, productividades de red,...). Si éstas pueden diferir debido a una suma de efectos aleatorios, podemos suponer que se distribuyen según una normal de media \bar{d}_{real} , que es el valor que buscamos. En ese caso, sabemos que

$$t_{exp} = \frac{\bar{d} - \bar{d}_{real}}{s/\sqrt{n}}$$

pertenece a la distribución t-Student con $n-1$ grados de libertad, siendo \bar{d} y s la media y la desviación típica muestrales, respectivamente.

Por tanto, hay un $(1-\alpha)*100\%$ de probabilidad de que el valor medio real \bar{d}_{real} se encuentre en el intervalo:

$$\bar{d} \pm \frac{s}{\sqrt{n}} t_{\alpha/2, n-1}$$

Utilidad: Podemos usar esta información para determinar un intervalo de confianza para \bar{d}_{real} , y no quedarnos simplemente con el valor medio muestral.

Ejemplo

Queremos determinar un intervalo de confianza del 95% para el tiempo medio de escritura de un determinado fichero en un disco duro. Para ello, se han realizado $n=8$ medidas experimentales:

#exp	d (ms)
1	835
2	798
3	823
4	803
5	834
6	825
7	813
8	829

$$\bar{d} = \frac{\sum_{i=1}^n d_i}{n} = 820ms$$

$$df = 8-1$$

$$s = \sqrt{\frac{\sum_{i=1}^n (d_i - \bar{d})^2}{n-1}} = 14ms$$

$$t_{\frac{\alpha}{2}, n-1} = t_{\frac{0,05}{2}, 8-1} = 2,36$$

$$\alpha = 0,05$$

df	0.20	0.10	0.05	0.02	0.01	0.001
1	3.0777	6.3138	12.7062	31.8205	63.6567	636.6192
2	1.8856	2.9200	4.3027	6.9646	9.9248	31.5991
3	1.6377	2.3534	3.1824	4.5407	5.8409	12.9240
4	1.5332	2.1318	2.7764	3.7469	4.6041	8.6103
5	1.4759	2.0150	2.5706	3.3649	4.0321	6.8688
6	1.4398	1.9432	2.4469	3.1427	3.7074	5.9588
7	1.4149	1.8946	2.3646	2.9980	3.4995	5.4079

O bien:

- En *Excel*, haciendo: `ABS(INV.T(alfa/2;n-1))`.
- En *Calc*, `DISTR.T.INV(alfa;n-1)`.

Por tanto, hay un 95% ($\alpha=0,05$) de probabilidad de que el tiempo medio de escritura **real** de ese fichero se encuentre en el intervalo:

$$\bar{d} \pm \frac{s}{\sqrt{n}} t_{\alpha/2, n-1} = 820 \pm \frac{14}{\sqrt{8}} \times 2,36 = [808, 832]ms$$

4.4 Diseño de experimentos de comparación de rendimiento

Planteamiento del problema

- Supongamos que queremos determinar cuáles de los siguientes factores afectan significativamente al rendimiento de un determinado servidor:
 1. Sistema Operativo: Windows Server, CentOS, Debian, Ubuntu.
 2. Memoria RAM: 32GB, 64GB, 128GB.
 3. Discos duros: SAS, SATA, IDE (=P-ATA).
- Y, en el caso de que afecten, cuál de los niveles del factor es significativamente mejor que el resto.
- ¿Qué experimentos debemos diseñar para ello y cómo debemos analizar los resultados?



Terminología

- **Variable respuesta o dependiente (=métrica):** El índice de rendimiento que usamos para las comparaciones. P.ej. tiempos de respuesta (R), productividades (X).
- **Factor:** Cada una de las *variables* que pueden afectar a la variable respuesta. P.ej. sistema operativo, tamaño de memoria, tipo de disco duro, tipo de procesador, número de microprocesadores, número de cores, tamaño de cada caché, compilador, algún parámetro configurable del S.O. o del servidor, etc.
- **Nivel:** Cada uno de los *valores* que puede asumir un factor. P.ej. para un S.O.: Windows, CentOS, Debian, Ubuntu; para un tipo de disco duro: SATA, IDE, SAS; para un parámetro del sistema operativo: ON, OFF, etc.
- **Interacción:** Una interacción ocurre cuando el efecto de un factor cambia para diferentes niveles de otro factor. P.ej. el hecho de usar un tipo determinado de S.O. puede afectar a cómo de importante sea usar una mayor cantidad de memoria DRAM.



Tipos de diseños experimentales

- **Diseños con un solo factor:** Se utiliza una configuración determinada como base y se estudia un factor cada vez, midiendo los resultados para cada uno de sus niveles. Problema: solo válida si descartamos que haya interacción entre factores.
 - **Diseños multi-factoriales completos:** Se prueba cada posible combinación de niveles para todos los factores. Ventaja: se analizan las interacciones entre todos los factores. Número total de combinaciones = $\prod_{i=1}^k n_i$. En nuestro ejemplo: 36 combinaciones.
 - **Diseños multi-factoriales fraccionados:** Término medio entre los anteriores. No todas las interacciones se verán reflejadas en los resultados, solo las de las interacciones que se consideren más probables.
- ☞ Todos ellos se pueden realizar con diferentes niveles de **repetición**: a) sin repeticiones, b) con todos los experimentos repetidos el mismo número de veces, c) con un número de repeticiones diferentes para cada nivel o cada factor.

Diseños con un solo factor

- **Ejemplo:** Para el servidor principal de nuestra empresa, queremos saber si la elección del tipo de disco duro afecta al rendimiento. Para ello, se ha escogido tres discos duros con tres interfaces distintas: **SAS**, **SATA** e **IDE** y se ha realizado un experimento que consiste en ejecutar, en condiciones reales y en presencia de aleatoriedad, un conjunto de programas usados habitualmente por el servidor y medir el **tiempo total de ejecución**. Este experimento se ha repetido **5 veces**:

#Exp.	SAS (s)	SATA (s)	IDE (s)
1	103	115	143
2	97	102	134
3	123	120	139
4	106	115	135
5	116	122	129
Medias	109.0	114.8	136.0
Efectos (ε_j)	-10.9	-5.1	16.1

$m_{\text{global}} = 119.9s$



¿Tiene influencia el factor “interfaz del disco duro” sobre el rendimiento? ¿Son las diferencias entre los discos duros significativas? → **Test ANOVA de un factor.**

Análisis de la Varianza (ANOVA) de un factor

$$\text{Modelo: } y_{ij} = m_{\text{global}} + \varepsilon_j + r_{ij} \quad i=1, \dots, n_{\text{rep}}; \quad j=1, \dots, n_{\text{niv}}$$

y_{ij} : Las observaciones. En nuestro caso los tiempos de ejecución obtenidos en cada prueba. El índice j recorre los distintos niveles del factor cuya influencia se quiere medir (en nuestro caso hay $n_{\text{niv}}=3$ niveles: SAS, SATA e IDE). El índice i recorre las distintas repeticiones para cada uno de esos niveles (en nuestro caso, $n_{\text{rep}}=5$ repeticiones).

m_{global} : Media global de todas las observaciones:
$$m_{\text{global}} = \frac{1}{n_{\text{rep}} \times n_{\text{niv}}} \sum_{i=1}^{n_{\text{rep}}} \sum_{j=1}^{n_{\text{niv}}} y_{ij}$$

ε_j : Efecto debido al nivel j -ésimo: $\varepsilon_j = \frac{1}{n_{\text{rep}}} \sum_{i=1}^{n_{\text{rep}}} y_{ij} - m_{\text{global}}$. Se cumple que $\sum_{j=1}^{n_{\text{niv}}} \varepsilon_j = 0$.

r_{ij} : Perturbaciones o error experimental (ruido). Deben cumplir:

- Que tengan varianza constante, independiente del nivel (si nº de exp./nivel no es homogéneo).
- Que su distribución sea normal.

Si no se cumplen, hay otros test alternativos: test de Kruskal-Wallis, test de Friedman.

☞ La principal pregunta que intenta contestar el test ANOVA es: ¿Tiene influencia el factor sobre la variable respuesta o, dicho de otro modo, algún ε_j es estadísticamente distinto de cero? Para ello, realiza la siguiente “hipótesis nula”: **Ho: El factor considerado no influye en el rendimiento.**

Análisis de la Varianza (ANOVA) de un factor (II)

- El objetivo del test ANOVA es contrastar esa **hipótesis (Ho)** de que el factor no influye en el **rendimiento** ($\varepsilon_j \approx 0 \forall j = 1 \dots n_{niv}$). Si esta hipótesis fuera cierta resulta que la siguiente medida experimental (estadístico):

$$F_{exp} \equiv \frac{SSA/(n_{niv} - 1)}{SSE/(n_{niv} \times (n_{rep} - 1))} \sim F_{n_{niv}-1, n_{niv} \times (n_{rep}-1)}$$

debería ser una muestra de una distribución F de Snedecor con $n_{niv}-1$ grados de libertad en el numerador y $n_{niv} \times (n_{rep}-1)$ en el denominador.

- SSA y SSE se calculan a partir de la siguiente descomposición de la varianza total de las muestras (SST, *Sum-of-Squares Total*):

$$SST = \sum_{i=1}^{n_{rep}} \sum_{j=1}^{n_{niv}} (y_{ij} - m_{global})^2 = n_{rep} \sum_{j=1}^{n_{niv}} (\varepsilon_j)^2 + \sum_{i=1}^{n_{rep}} \sum_{j=1}^{n_{niv}} (r_{ij})^2 = SSA + SSE$$

- SSA = Varianza explicada por los efectos o alternativas (*Sum-of-Squares Alternatives*).
- SSE = Varianza residual o del error (*Sum-of-Squares Error*).

Análisis de la Varianza (ANOVA) de un factor (III)

Para nuestro ejemplo concreto:

$$SST = 2809$$

$$SSA = 2020$$

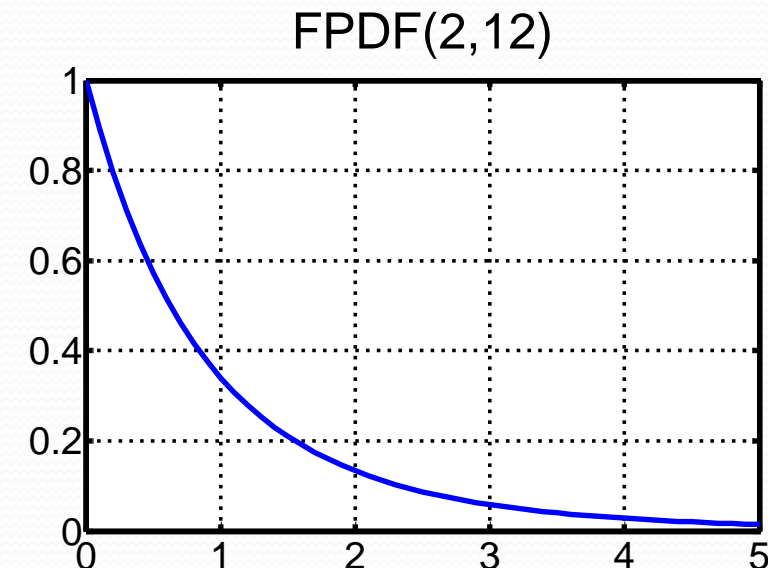
$$SSE = 789$$

$$F_{exp} \equiv \frac{\frac{SSA}{n_{niv} - 1}}{\frac{SSE}{(n_{niv} \times (n_{rep} - 1))}} = \frac{\frac{2020}{3 - 1}}{\frac{789}{(3 \times (5 - 1))}} = 15,37$$

¿Qué probabilidad hay de que la muestra 15,37 o superior se haya extraído de una distribución $F_{2,12}$?
 $P - value = P(F \geq 15,37; 2, 12) = 0,00049$. Con *Excel* y *Calc*: $DISTR.F(15,37;2;12) = 0,00049$.

Identifico ese valor como la probabilidad de que la hipótesis (H_0) de que el factor no influye pueda ser cierta.

- Si la probabilidad es menor que $\alpha=0,05$ diremos que *descartamos la hipótesis de que el factor no influya* a un $(1-\alpha) \times 100\% = 95\%$ de confianza.
- Si el factor influye, a continuación (análisis *post-hoc*) comparamos las medias de cada nivel unas con otras usando esencialmente un *test t* que aquí se llama: prueba de múltiples rangos o de comparaciones múltiples.



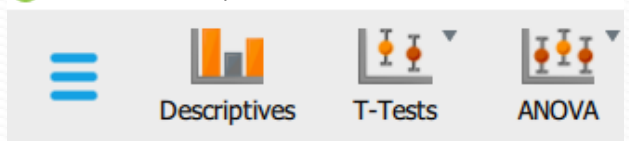
En resumen: Test ANOVA de un factor

#Experimento	Rendimiento nivel 1	Rendimiento nivel 2	...	Rendimiento nivel n_{niv}
1	y_{11}	y_{12}		$y_{1n_{niv}}$
2	y_{21}	y_{22}		$y_{2n_{niv}}$
...
n_{rep}	$y_{n_{rep}1}$	$y_{n_{rep}2}$		$y_{n_{rep}n_{niv}}$

- **Modelo:** $y_{ij} = \mathbf{m}_{global} + \varepsilon_j + \mathbf{r}_{ij}$ $i=1,...,n_{rep}; j=1,...,n_{niv}$
- H_0 : Rendimiento de todos los niveles del factor es equivalente ($\varepsilon_j=0, j=1,...,n_{niv}$) \equiv El factor no influye en el rendimiento.
- Se calcula $F_{exp} \equiv \frac{SSA/(n_{niv}-1)}{SSE/(n_{niv} \times (n_{rep}-1))} \sim F_{n_{niv}-1, n_{niv} \times (n_{rep}-1)}$
- p-value \approx Prob (Ho podría ser cierta).
- Rechazamos H_0 para un nivel de confianza $(1- \alpha)*100(\%)$ si valor-p< α . En ese caso, el siguiente paso será comparar las medias de cada nivel unas con otras usando un test t: *prueba de múltiples rangos o de comparaciones múltiples*.

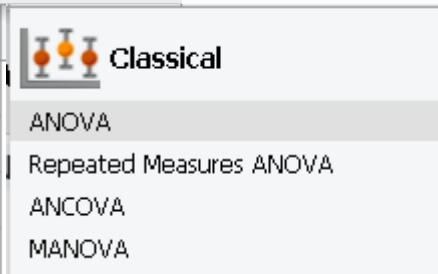
ANOVA de un factor con JASP (jasp-stats.org)

Anova1Factor (D:\misdoc\docencia\miISE\Slides\ANOV

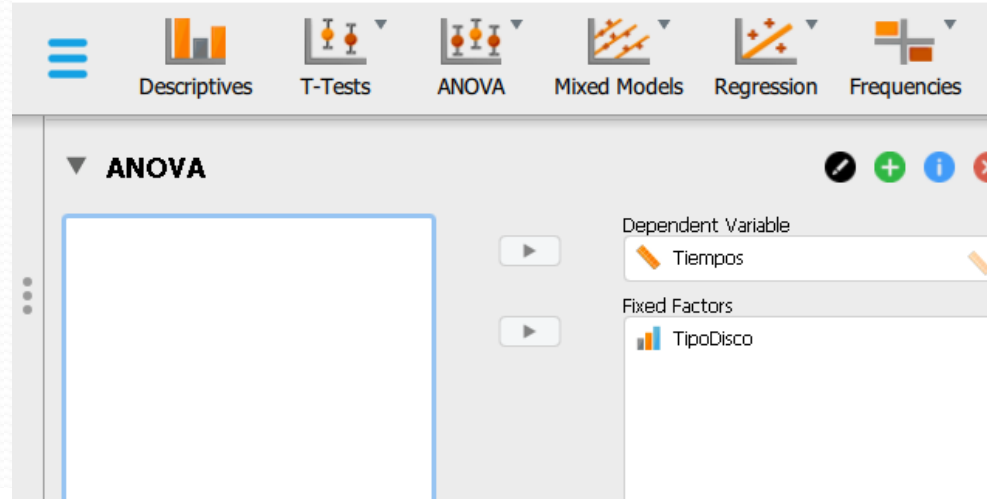


1: SAS
2: SATA
3: IDE

	TipoDisco	Tiempos
1	SAS	103
2	SAS	97
3	SAS	123
4	SAS	106
5	SAS	116
6	SATA	115
7	SATA	102
8	SATA	120
9	SATA	115
10	SATA	122
11	IDE	143
12	IDE	134
13	IDE	139
14	IDE	135
15	IDE	129



Anova1Factor* (D:\misdoc\docencia\miISE\Slides\ANOVA)



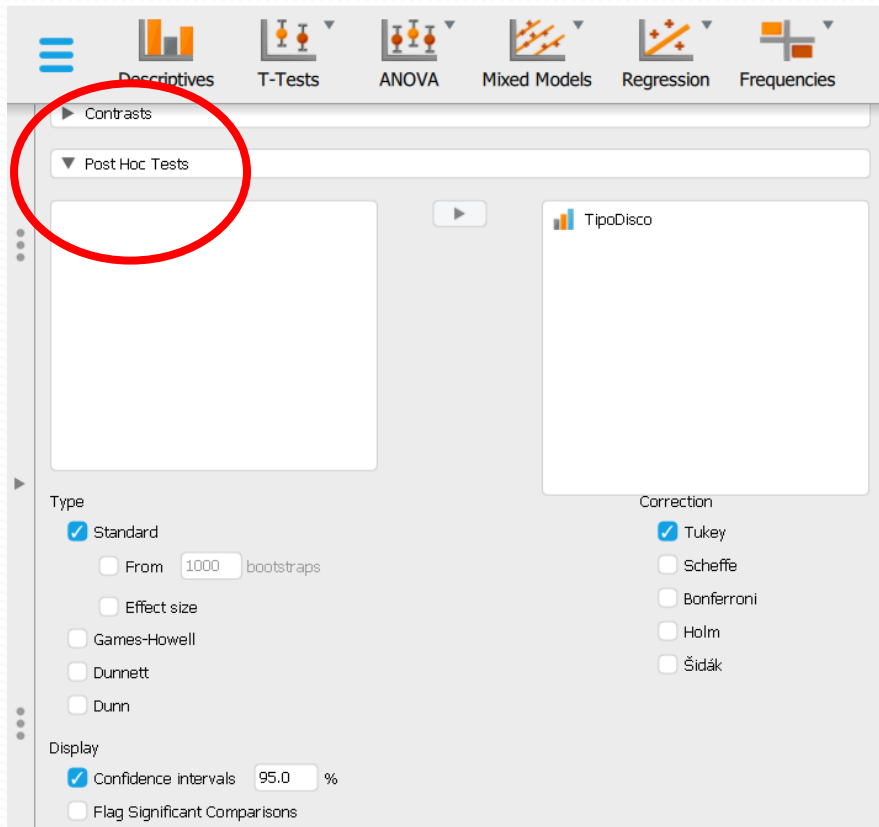
ANOVA - Tiempos

Cases	Sum of Squares	df	Mean Square	F	p
TipoDisco	2020.133	2	1010.067	15.366	4.904e -4
Residuals	788.800	12	65.733		

Rechazaremos la hipótesis H_0 para $\alpha = 0,05$ (al menos). Concluimos que para un nivel de confianza del 95%, la interfaz de disco duro afecta significativamente al rendimiento del equipo.

ANOVA de un factor con JASP (post-hoc test)

Como el factor influye, hacemos ahora un test t entre cada combinación de niveles para comparar el efecto en el rendimiento de cada tipo de disco duro: **prueba de múltiples rangos o de comparaciones múltiples**.



Intervalo de confianza (95%) para $\overline{d_{real}}$				p-value de cada test t realizado		
95% CI for Mean Difference						
	Mean Difference	Lower	Upper	SE	t	Ptukey
SAS \rightarrow SATA	-5.800	-19.480	7.880	5.128	-1.131	0.514
SAS \rightarrow IDE	-27.000	-40.680	-13.320	5.128	-5.266	5.399e-4
SATA \rightarrow IDE	-21.200	-34.880	-7.520	5.128	-4.134	0.004

- La primera fila de la tabla es un test t entre SAS y SATA, la segunda entre SAS de IDE y la tercera entre SATA e IDE.
- La última columna es el p-value de cada test t realizado.
- Concluimos que, al 95% de confianza, el disco IDE es claramente peor que los otros dos, pero que las diferencias entre SAS y SATA, para este problema, no son estadísticamente significativas.

ANOVA de dos factores con JASP

Dependent Variable
Tiempos

Fixed Factors
TipoDisco
S.O.

Anova2Factor* (D:\misdoc\docencia\miISE\Slides\

	TipoDisco	S.O.	Tiempos
1	SAS	WSERVER	109
2	SAS	WSERVER	110
3	SAS	CENTOS	110
4	SAS	CENTOS	115
5	SAS	DEBIAN	108
6	SAS	DEBIAN	109
7	SAS	UBUNTU	110
8	SAS	UBUNTU	108
9	SATA	WSERVER	110
10	SATA	WSERVER	112
11	SATA	CENTOS	110
12	SATA	CENTOS	111
13	SATA	DEBIAN	111
14	SATA	DEBIAN	109

- Ahora, consideraremos dos factores: el *Tipo de Disco* (SAS, SATA, IDE) y el *S.O.* (Windows Server, CentOS, Debian, Ubuntu).
- Esencialmente realizamos varios test ANOVA en paralelo. Para el primero de ellos, la hipótesis nula es que el factor *Tipo de Disco* no influye. Para el segundo que el factor *S.O.* no influye. Para el tercero, que el factor “*Tipo de Disco* * *S.O.*” no influye, es decir, que no hay interacción entre esos dos factores.

Cases	Sum of Squares	df	Mean Square	F	p
TipoDisco	160.333	2	80.167	21.143	1.167e -4
S.O.	12.458	3	4.153	1.095	0.389
TipoDisco * S.O.	44.667	6	7.444	1.963	0.151
Residuals	45.500	12	3.792		

- Concluimos que, al 95% de confianza, el tipo de disco influye significativamente en los tiempos medidos. Sin embargo, no podemos rechazar las hipótesis de que el *S.O.* no influye en dichos tiempos y que no existe interacción entre el tipo de disco y el *S.O.* Por tanto, solo debemos preocuparnos en hacer el *post-hoc test* para el factor *Tipo de Disco*.