Performance Evaluation of NFS over a Wide-Area Network

Using **D**esign of **E**xperiments methods

Abdulqawi Saif
abdulqawi.saif@loria.fr

Lucas Nussbaum
lucas.nussbaum@loria.fr

July 6, 2016
COMPAS'2016 - Lorient, France

Big Data

- Changes the way we store data:
  - Local network storage ⇒ remote, centralized storage *(e.g: clusters)*
  - Storage arrays ⇒ Scalable, flexible storage solutions (Ceph, GlusterFS, ...)

Big Data

- Changes the way we store data:
  - Local network storage ⇒ remote, centralized storage *(e.g: clusters)*
  - Storage arrays ⇒ Scalable, flexible storage solutions (Ceph, GlusterFS, ...)

Software-defined solutions

- Most are object storage systems
  - Provide GET/PUT methods *(perform well over high latency network)*
  - Don't provide a POSIX-interface! *(still used by several applications)*

Big Data

- Changes the way we store data:
  – Local network storage ⇒ remote, centralized storage *(e.g: clusters)*
  – Storage arrays ⇒ Scalable, flexible storage solutions (Ceph, GlusterFS, ...)

Software-defined solutions

- Most are object storage systems
  - Provide GET/PUT methods *(perform well over high latency network)*
  - Don't provide a POSIX-interface! *(still used by several applications)*

To overcome this issue

- Kernel modification on clients *(Compatibility impacts)*
- Using FUSE filesystem *(Performance impacts)*
- Still providing NFS or CIFS interfaces ✓

Big Data

- Changes the way we store data:
  – Local network storage ⇒ remote, centralized storage *(e.g: clusters)*
  – Storage arrays ⇒ Scalable, flexible storage solutions (Ceph, GlusterFS, ...)

Software-defined solutions

- Most are object storage systems
  - Provide GET/PUT methods *(perform well over high latency network)*
  - Don't provide a POSIX-interface! *(still used by several applications)*

To overcome this issue

- Kernel modification on clients *(Compatibility impacts)*
- Using FUSE filesystem *(Performance impacts)*
- Still providing NFS or CIFS interfaces ✓ *(NFS performance?)*

Network File System (NFS)

- Several versions over time since 1984
- Sharing files in a network over a heterogeneous machines [1]
- NFSv3 is the mostly used and NFSv4 is the latest version [2,3]

NFS performance is influenced by :

- NFS server-side configurations
- Network characteristics *(latency, throughput, ...)* ✓
- NFS client-side configurations (tunings) ✓

[1] Sandberg (R.), Goldberg (D.), Kleiman (S.), Walsh (D.) et Lyon (B.). – Design and imple- mentation of the sun network filesystem. – In Proceedings of the Summer USENIX conference, pp. 119–130, 1985.
[2] Shepler (S.), Eisler (M.), Robinson (D.), Callaghan (B.), Thurlow (R.), Noveck (D.) et Beame (C.). – Network file system (nfs) version 4 protocol. Network, 2003.
[3] Tarasov (V.), Hildebrand (D.), Kuenning (G.) et Zadok (E.). – Virtual machine workloads : The case for new nas benchmarks. – In Presented as part of the 11th USENIX Conference on File and Storage Technologies (FAST 13), pp. 307–320, 2013.

Related work

Several researches evaluate NFS performance:

- Client configurations (tunings) have a major impact on NFS performance [4]
- Performance of NFSv3 and NFSv4 depends on the network latency[1]
  - NFSv4 is faster than NFSv3 on a high latency network
  - NFSv3 is faster than NFSv4 on a low latency network
- NFSv3 can tolerate high latency more than NFSv2 [3]
- Comparable performance between NFS & other protocols such as iSCSI [2]
- ...

[1] Chen (M.), Hildebrand (D.), Kuenning (G.), Shankaranarayana (S.), Singh (B.) et Zadok (E.). – Newer is sometimes better : An evaluation of nfsv4. 1. – In Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems, pp. 165–176. ACM, 2015.
[2] Radkov (P.), Yin (L.), Goyal (P.), Sarkar (P.) et Shenoy (P. J.). – A performance comparison of nfs and iscsi for ip-networked storage. – In FAST, pp. 101–114, 2004.
[3] Martin (R. P.) et Culler (D. E.). – Nfs sensitivity to high performance networks. ACM SIG- METRICS Performance Evaluation Review, vol. 27, n1, 1999, pp. 71–82.
[4] Ou (Z.), Hwang (Z.-H.), Ylä-Jääski (A.), Chen (F.) et Wang (R.). – Is cloud storage ready ? a comprehensive study of ip-based storage systems. UCC15, 2015.

Research motivations

NFS performance evaluation

- How will it behave on a realistic network ?
  - ⇒ Most of related works use emulated networks
- Is it efficient to be used on a high latency environment ?

Using statistical methods

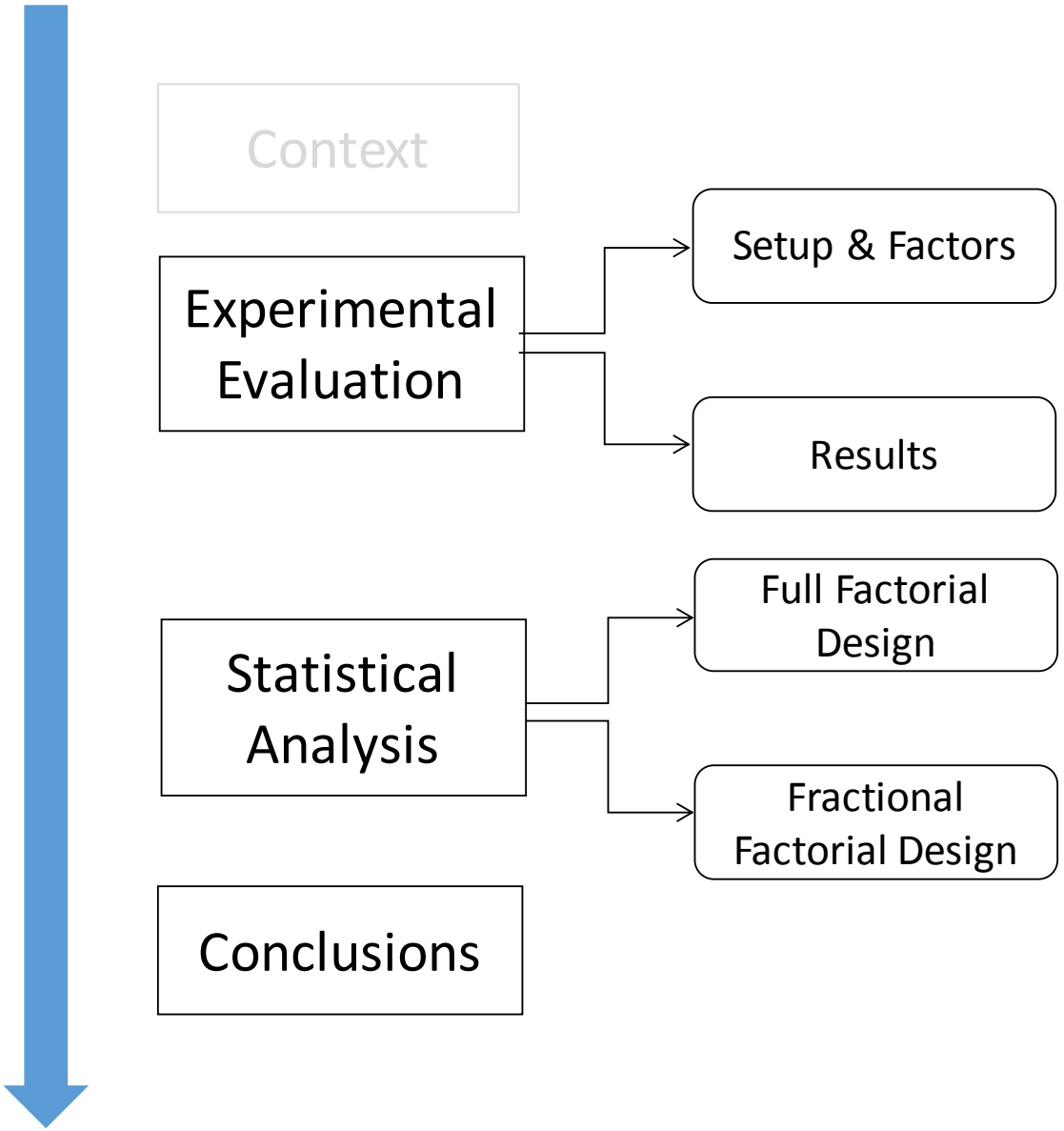
which are infrequently used in Computer Science community

- Several surveys (1994 [1], 1998 [2], 2009 [3]) on hundreds papers of ACM
  - Many papers have no experimental validation at all
  - 40%-50% of papers that require an experimental validation had none
- A study on Europar conference papers, made by E. Jeannot

Year	Total papers	With error bars	Percentage
2007	89	5	5.6
2008	89	3	3.4
2009	86	2	2.4
2010	90	6	6.7
2011	81	7	8.6
2007-2011	**435**	**23**	**5.3**

[1] Paul Lukowicz et al. "Experimental Evaluation in Computer Science : A Quantitative Study" In : Journal of Systems and Software 28 (1994).
[2] M.V. Zelkowitz and D.R. Wallace. "Experimental models for validating technology". In : Computer 31.5 (May 1998).
[3] Marvin V. Zelkowitz. "An update to experimental models for validating computer technology". In : J. Syst. Softw. 82.3 (Mar. 2009).

Outline


Context
Experimental Evaluation
Setup & Factors
Results
Statistical Analysis
Full Factorial Design
Fractional Factorial Design
Conclusions

Experimental Evaluation

Experimental goals:

- ⇒ Tuning NFS with several parameters
- ⇒ Reading, writing evaluation using several files
- ⇒ Using several categories of latency

Experimental Setup:

- ⇒ 4 machines, different sites
- ⇒ Debian Jessie & Linux 3.16.0
- ⇒ Cleaning cache before each operation
- ⇒ Sequential operations

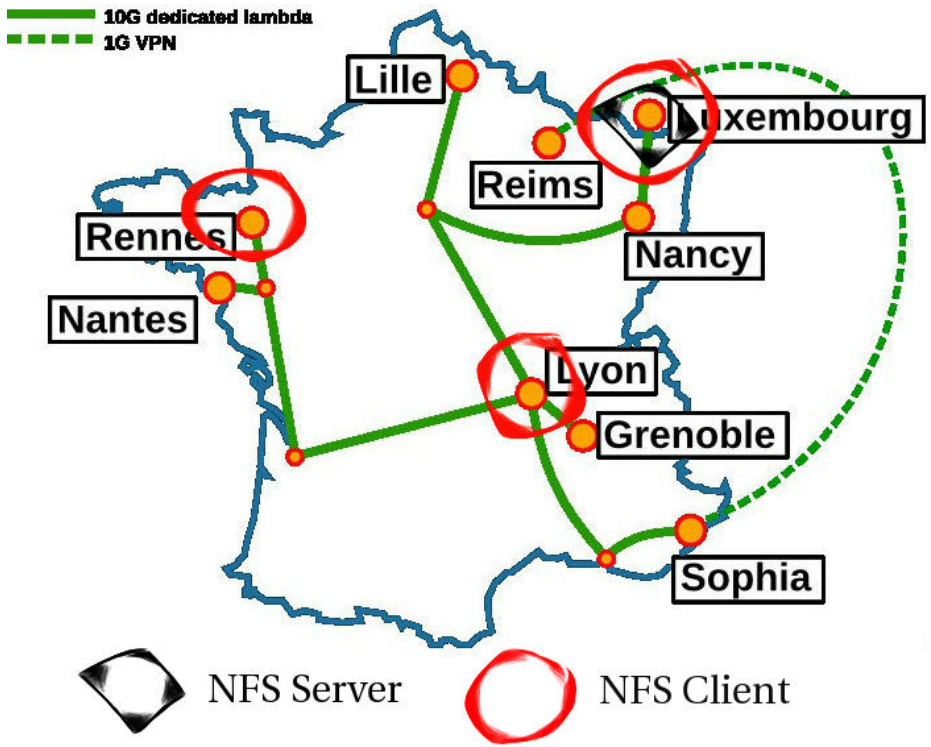


FIGURE – Implemented topology on Grid5000 testbed

Experimental Evaluation of NFS

Experimental factors:

- NFS versions
  - NFSv3
  - NFSv4
- Synchronicity
  - Sync mode
  - Async mode

Experimental Evaluation of NFS

Experimental factors:

- NFS versions
  - NFSv3
  - NFSv4
- Synchronicity
  - Sync mode
  - Async mode
- NFS I/O Size
  - 64KB
  - 1MB
- Storage Type
  - HDD (ext4)
  - In-memory (tmpfs)

Experimental Evaluation of NFS

Experimental factors:

- NFS versions
  - NFSv3
  - NFSv4
- NFS I/O Size
  - 64KB
  - 1MB
- Transferred files
  - 100MB
  - 5GB
- Synchronicity
  - Sync mode
  - Async mode
- Storage Type
  - HDD (ext4)
  - In-memory (tmpfs)
- Average latencies
  - 0.027 ms
  - 6.87 ms
  - 13.9 ms

Experimental methodology:

⇒ All combinations of factors

⇒ Against reading and writing

⇒ Each executed 5 times

⇒ $96 * 2 * 5 = 960$ executions

Experimental Evaluation of NFS - Results

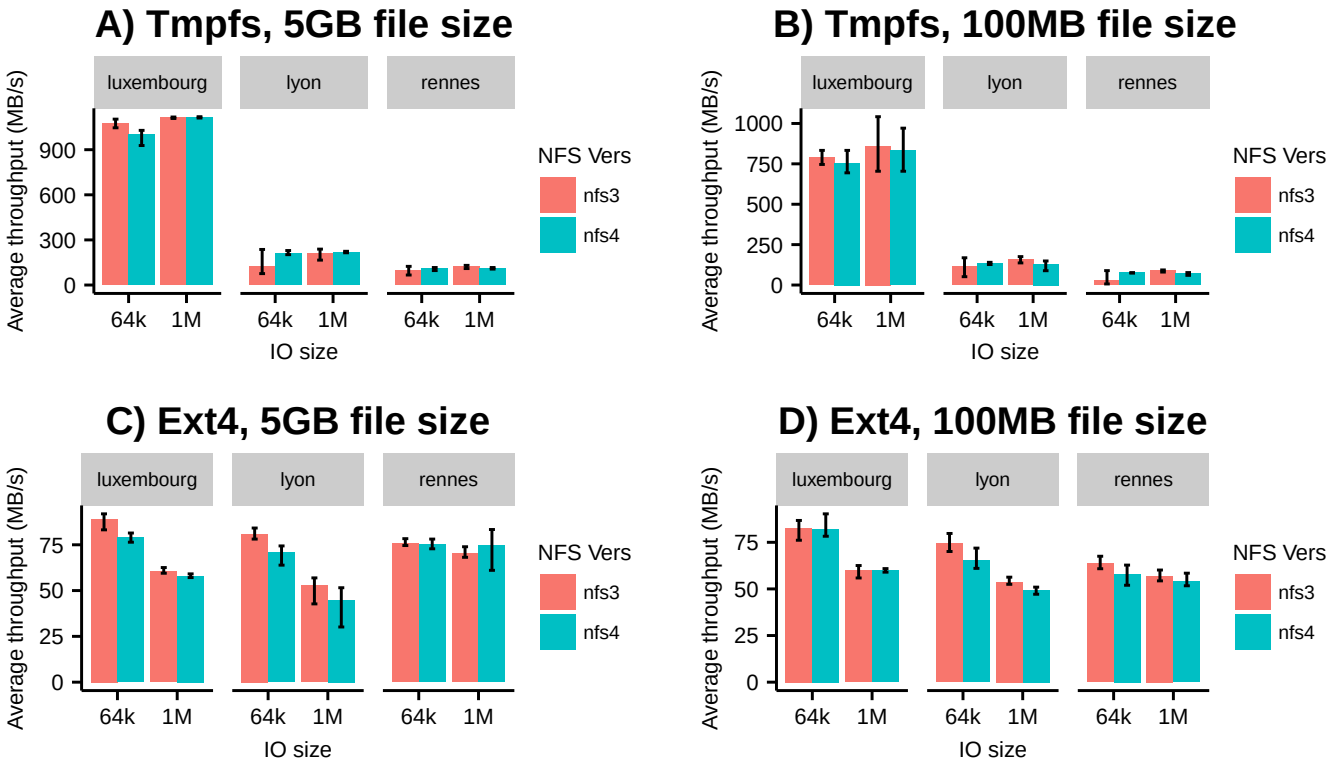


FIGURE – Reading results with *Sync-on*

- Lower speed of *HDD* impacts the performance
- NFS versions have almost the same performance
- The impact of latency is clearly shown

Experimental Evaluation of NFS - Results

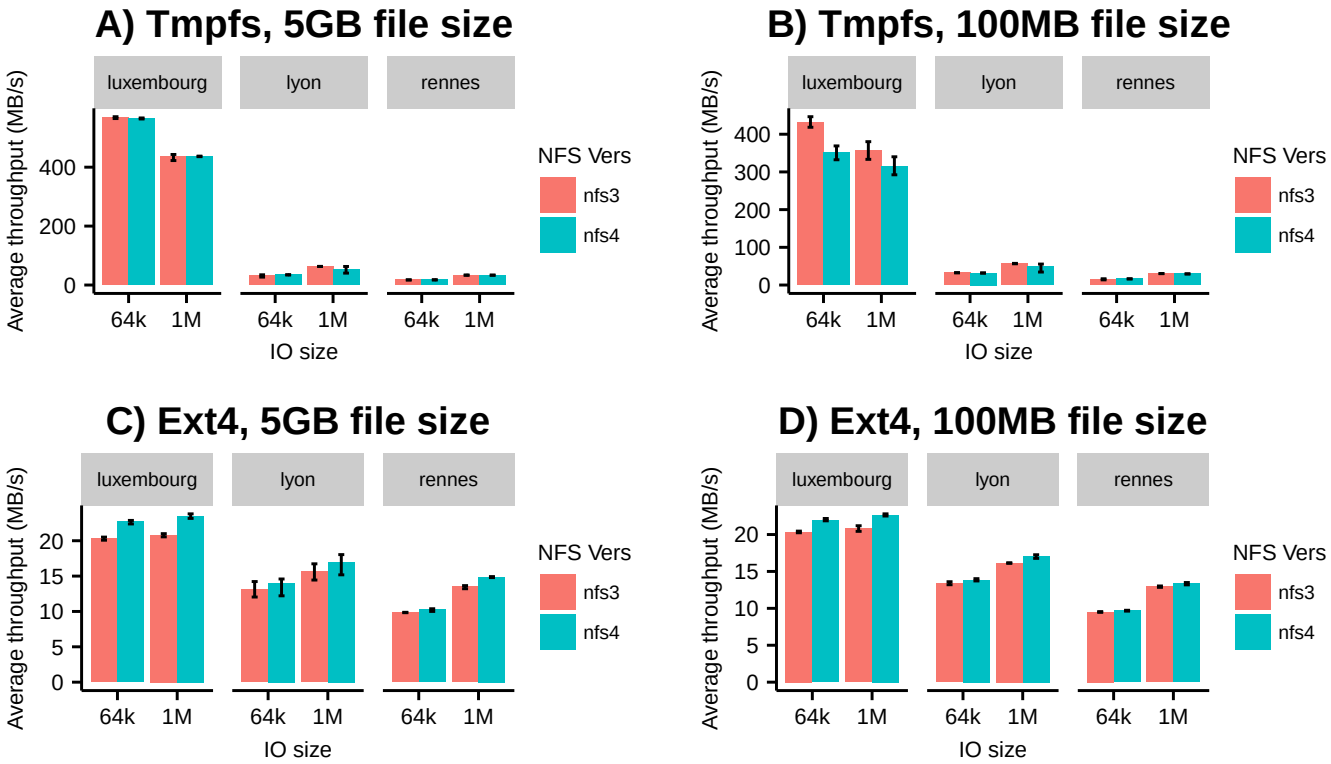


FIGURE – Writing results with *Sync-on*

- HDD limitation also appears here
- Clients performance is also affected by the latency
- NFS IO size on local client affects the performance

Experimental Evaluation of NFS - Results

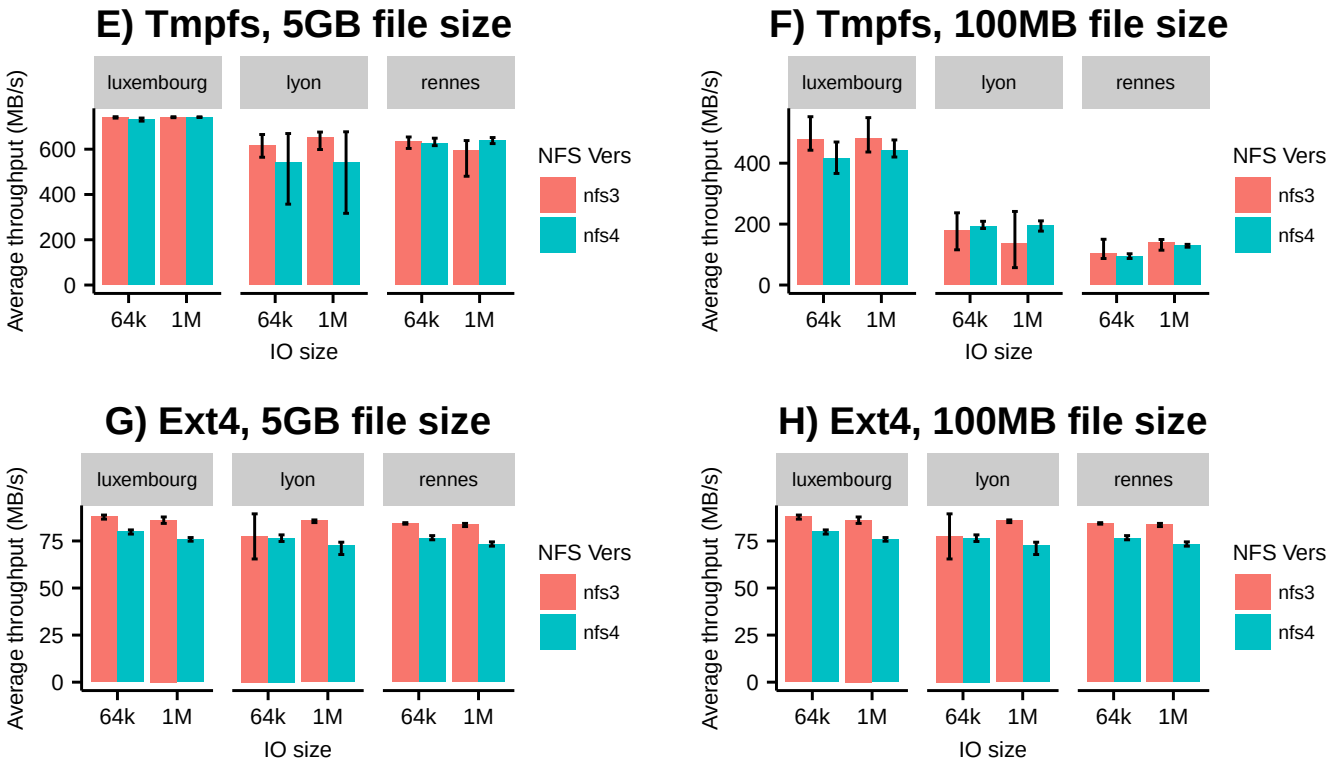


FIGURE – Writing results with *Sync-off*

- Better than the Sync results
- Latency's impact is mitigated (see Lyon & Rennes clients)
- Trade-offs Reliability Vs Performance

Statistical Analysis

Statistical analysis

Goals

Using statistical methods leads to:

- Exclude the impact of chance and unknown factors
- Deeper analysis of results
- Results validation
- Determine the impacts of factors on the results

Goals

Using statistical methods leads to:

- Exclude the impact of chance and unknown factors
- Deeper analysis of results
- Results validation
- Determine the impacts of factors on the results

Full factorial analysis

- All combinations of factors (I x J x K...)
- Include analyzing all interactions *(effect of one factor depends on another factor's levels)*
- Useful at the beginning of experiments
  - First pass over all factors
  - Exclude less-impact factors
  - Focusing on the remaining ones

X_1	X_2	X_3	X_1X_2	X_1X_3	X_2X_3	$X_1X_2X_3$
+	+	+	+	+	+	+
+	+	-	+	-	-	-
+	-	+	-	+	-	-
+	-	-	-	-	+	+
-	+	+	-	-	+	-
-	+	-	-	+	-	+
-	-	+	+	-	-	+
-	-	-	+	+	+	-

TABLE – 2^3 : all combination of factors and interactions

Full factorial design

Factors	Low level (-1)	High level (+1)
NFS version	NFSv3	NFSv4
Storage Type	In-memory(Tmpfs)	HDD(Ext4)
Syncro.	Sync	Async
NFS IO Size	64KB	1MB
File size	50MB	5GB
Latency	0.027 ms (Lux.)	13.9 ms (Rennes)

TABLE – The involved factors in Full factorial design analysisl

⇒ 2^6 requires analyzing 58 interactions beside the factors

⇒ Several operations to obtain the effects of each factor & theirs interactions

- mean, standard deviation, variations, sum of squares,...

⇒ *Predicted model behavior Vs obtained behavior* ⇒ model errors

Full factorial design

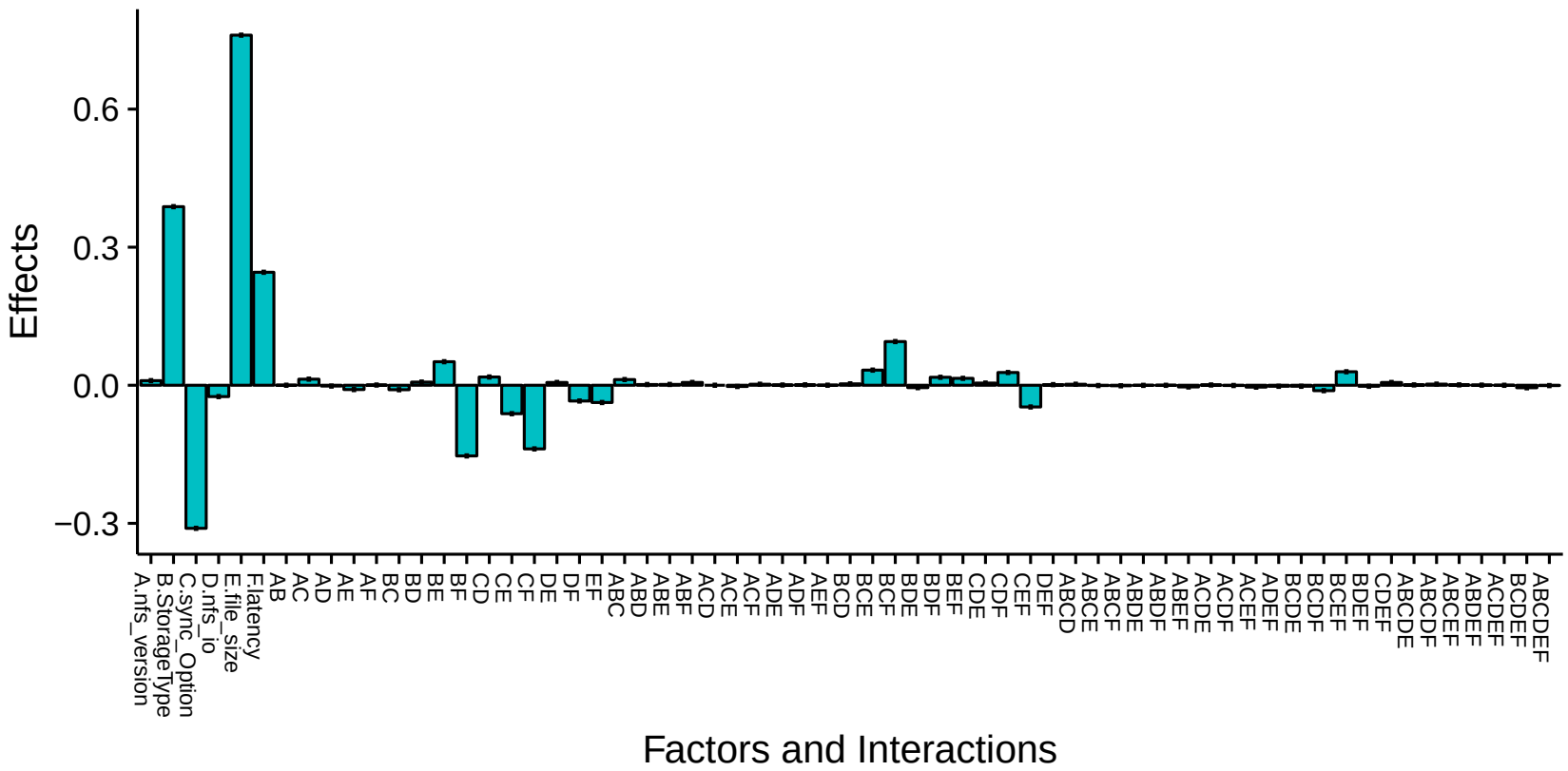


FIGURE – Full factorial design results

⇒ Effects: to which degree a factor or an interaction explains the results

- The higher the absolute value, the higher impact on results

Full factorial design

⇒ Exponential growth in experimental size!

- 15 factors with 2 levels (2^{15}) requires 32768 experiments!
- 15 factors with 2 levels (2^{15}) will analyze 32753 interactions!

⇒ Is it useful to analyze all interactions ? Specifically when:

- No complex interaction between factors
- Main effects estimated to be obtained from the factors

Fractional factorial design

Key idea

- Reduce the number of experiments by sacrificing the effects of interactions

How ?

Require a fraction of full factorial design experiments: 2^{K-P}

- Which level of abstraction : 70%, 20%, ...? *(Researcher has the choice)*

But, how to choose the best fraction (runs)?

$\Rightarrow$ Best 4 runs of 2^{3-1} design ?

X_1	X_2	X_3
-	-	+
+	-	+
-	+	+
+	+	+

TABLE – Variation of X3?

Fractional factorial design

Key idea

- Reduce the number of experiments by sacrificing the effects of interactions

How ?

Require a fraction of full factorial design experiments: 2^{K-P}

- Which level of abstraction : 70%, 20%, ...? *(Researcher has the choice)*

But, how to choose the best fraction (runs)?

$\Rightarrow$ Best 4 runs of 2^{3-1} design ?

X_1	X_2	X_3
-	-	+
+	-	+
-	+	+
+	+	+

TABLE – Variation of X3?

X_1	X_2	X_3
-	-	-
+	-	+
-	+	-
-	+	+

TABLE – X1 varies equally?

Fractional factorial design

Key idea

- Reduce the number of experiments by sacrificing the effects of interactions

How ?

Require a fraction of full factorial design experiments: 2^{K-P}

- Which level of abstraction : 70%, 20%, ...? *(Researcher has the choice)*

But, how to choose the best fraction (runs)?

$\Rightarrow$ Best 4 runs of 2^{3-1} design ?

X_1	X_2	X_3
-	-	+
+	-	+
-	+	+
+	+	+

TABLE – Variation of X3?

X_1	X_2	X_3
-	-	-
+	-	+
-	+	-
-	+	+

TABLE – X1 varies equally?

X_1	X_2	X_3
+	-	-
-	-	+
-	+	-
+	+	+

TABLE – Perfect!

Fractional factorial design

Factors	Low level (-1)	High level (+1)
(A) NFS version	NFSv3	NFSv4
(B) Storage Type	In-memory(Tmpfs)	HDD(Ext4)
(C) Syncro.	Sync	Async
(D) NFS IO Size	64KB	1MB
(E) File size	50MB	5GB
(F) Latency	0.027 ms (Lux.)	13.9 ms (Rennes)

TABLE – The involved factors in fractional design

NFS study & 2^{k-p}

- A reminder: 6 factors
- We choose $P = 3$
- 8 configurations are involved
- Factors: A, B & C used as main factors

Fractional factorial design

Factors	Low level (-1)	High level (+1)
(A) NFS version	NFSv3	NFSv4
(B) Storage Type	In-memory(Tmpfs)	HDD(Ext4)
(C) Syncro.	Sync	Async
(D) NFS IO Size	64KB	1MB
(E) File size	50MB	5GB
(F) Latency	0.027 ms (Lux.)	13.9 ms (Rennes)

TABLE – The involved factors in fractional design

A	B	C	D_{AB}	E_{AC}	F_{BC}	ABC
-1	-1	-1	1	1	1	-1
-1	-1	1	1	-1	-1	1
-1	1	-1	-1	1	-1	1
-1	1	1	-1	-1	1	-1
1	-1	-1	-1	-1	1	1
1	-1	1	-1	1	-1	-1
1	1	-1	1	-1	-1	-1
1	1	1	1	1	1	1

TABLE – The involved configurations

NFS study & 2^{k-p}

- A reminder: 6 factors
- We choose $P = 3$
- 8 configurations are involved
- Factors: A, B & C used as main factors

Note that :

- The first row represents: NFSv3, Tmpfs, Sync, 1MB as IO, 5GB file on Rennes client
- Three confounding factors are designed
  - Vary according to the design

Fractional factorial design

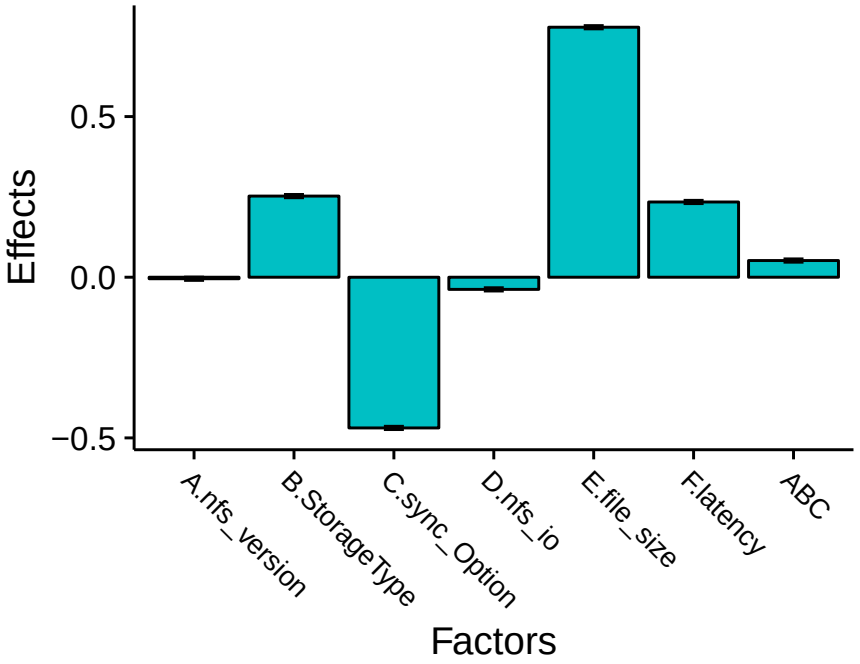


FIGURE – Fractional design results

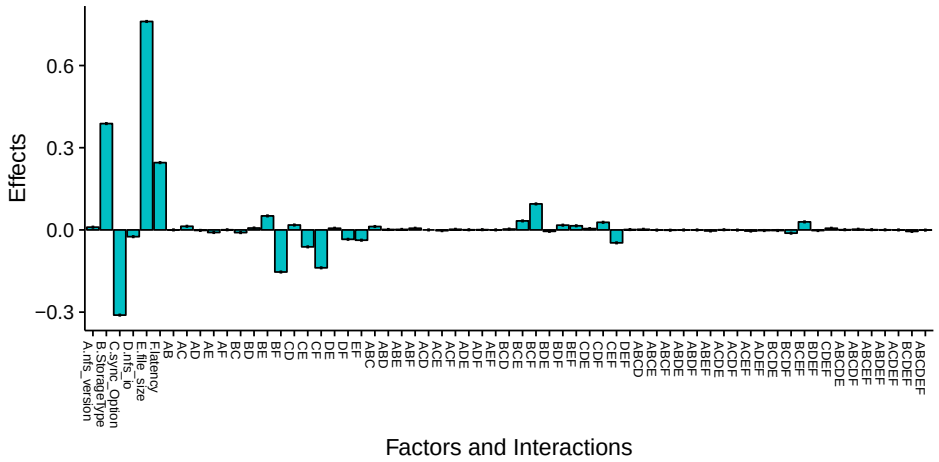


FIGURE – Full factorial design results

Fractional factorial design

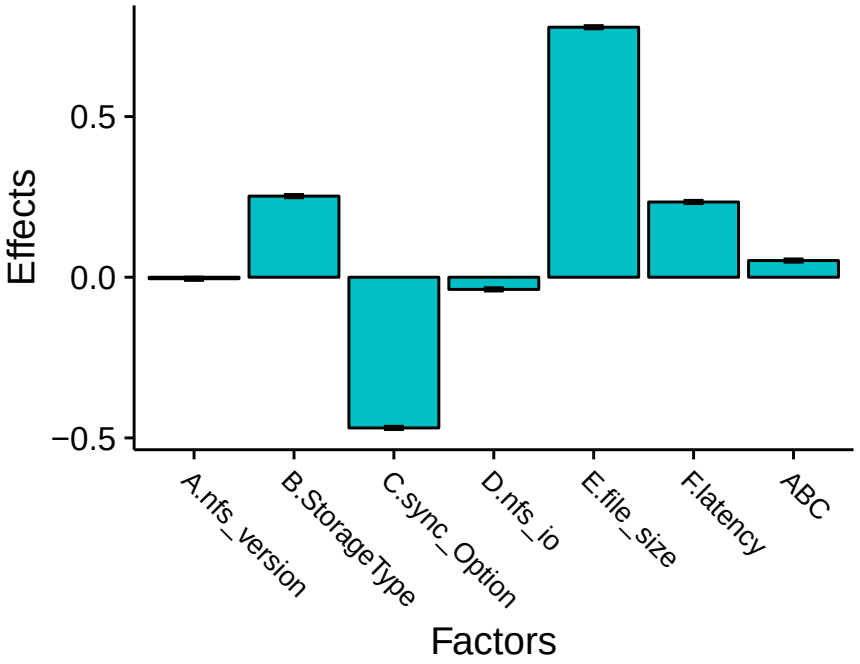


FIGURE – Fractional design results

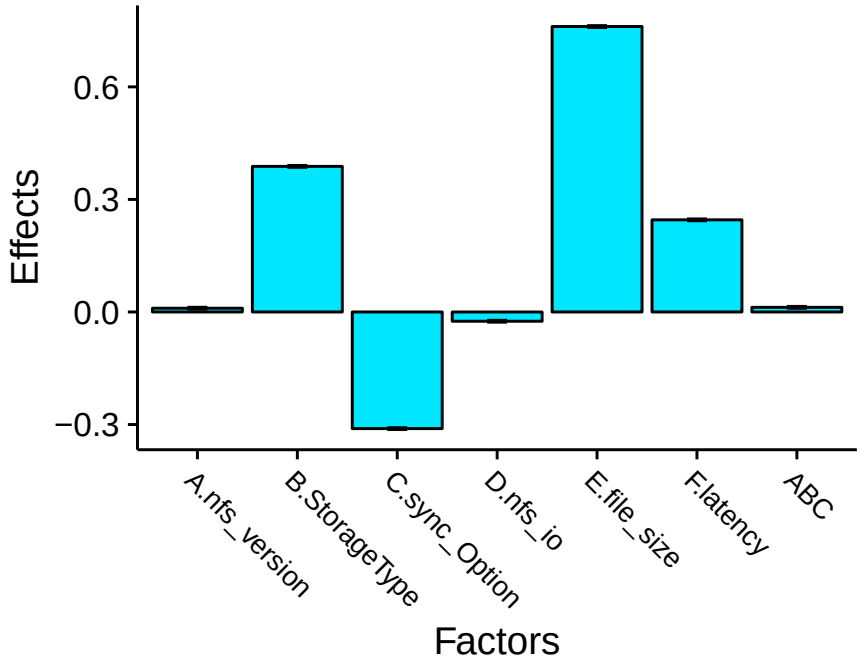


FIGURE – Full factorial design results - Factors

Conclusions

⇒ NFS tunings allow us to mitigate the effects of latency

⇒ Statistical methods are mainly important to well understand the results

⇒ Fractional design reduces the number of experiments

- In this study: 90 instead of 640 experiments required by full factorial experiments!

Future work :

⇒ First work on NFS, can be extended to cover several protocols, parameters

Conclusions

⇒ NFS tunings allow us to mitigate the effects of latency

⇒ Statistical methods are mainly important to well understand the results

⇒ Fractional design reduces the number of experiments

- In this study: 90 instead of 640 experiments required by full factorial experiments!

Future work :

⇒ First work on NFS, can be extended to cover several protocols, parameters

Any Questions?