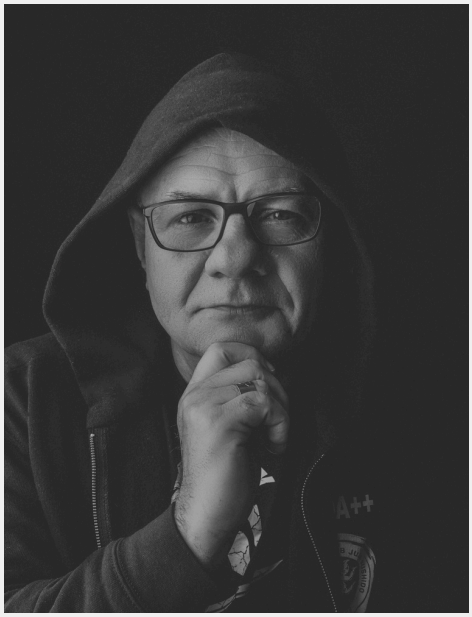
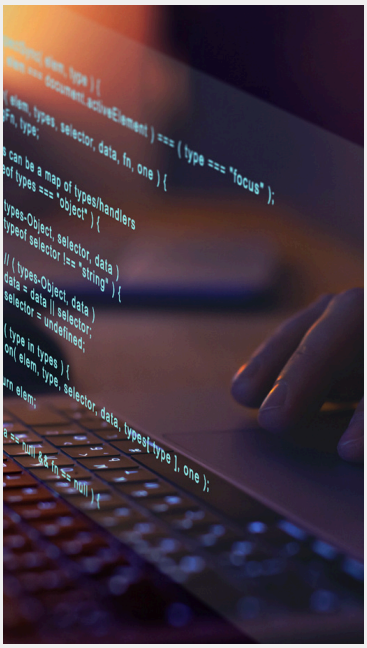
Requiem for the VFP ?
I don't think so.

13 years ago in 2012, software created by me used to work in bakery segments in large businesses.
The origin of this software goes back to the year 1998, it was written in C++ language and Clipper 5.2E.

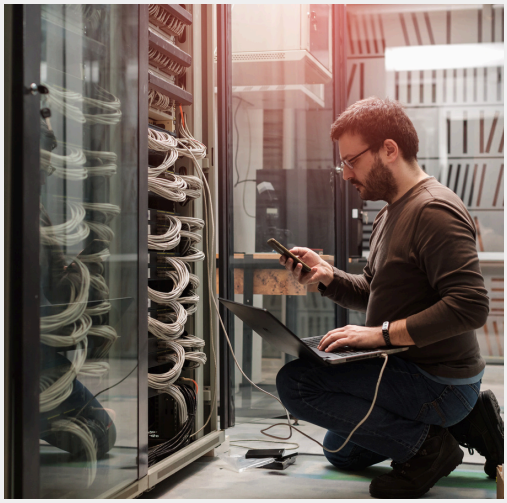


Back in those days, the only viable (I kept the majority of the source code from Clipper) alternative for me for transitioning into the 32bit world was obviously Visual FoxPro by Microsoft.
The language was basically the same as Clipper, with a few minor changes. I quickly learned the capabilities of this tool in terms of efficiency. I soon realized why the company M$ bought Ashton-Tate FoxPro in 1992.
They bought the very advanced math within its motor, which we can now admire in new versions of MSSQL.



Because there was no MSQL in 1998 and SQL 7 by Microsoft (do you still remember that ?) only made me feel pity when I tested its efficiency, I decided on VFP and DBF/CDX files. Back then the barrier of 2GB for the size of the file was a significant problem for me, however, I was hoping that soon enough the M$ company would break this and other barriers, which were gripping the VFP at the time (the lack of possibility in building native services, no built-in multithreading, etc.). I was hoping that the VFP company would remove those barriers soon. As we know, they did not, and killed the product in 2007 along with many others that they bought just to intercept the technology and end the product itself to pretend that this engineering achievement belongs to them. The inconveniences mentioned above were resolved, naturally, by using the libraries written in C++ i.e., vfp2c32.
source: **ChristianEhlscheid** https://github.com/ChristianEhlscheid/vfp2c32
In 2012, I reached the limit of my technological capabilities in terms of DBF/CDX files. My biggest client had about 70 users around that time working in headquarters on desktop-type applications. As well as over 300 different positions in the WAN network, mainly POS, which contacts the headquarters several times per hour.
Upon reviewing the programming languages present in the market at that time, I came across a compiler by a German company, Alaska Software | Xbase++.

https://www.alaska-software.com/



Language very similar (the same XBASE language family) to VFP, it has many advantages, but I will only mention some of them.

- ✓ is very fast and stable (Alaska Software is a direct overlay for c++).
- ✓ outstanding multithreading.
- ✓ very strong native support for TCP/IP, a set of functions that allows building any TCP/IP and UDP protocols.
- ✓ outstanding cooperation with COM/DCOM VFP servers.
- ✓ a great support for Postgresql.

Using this tool, I created the TCP/IP protocol, which executes VFP's every command on the side of the server.
So client-server for DBF/CDX files and much more. Because at present it can only service the following database engines:

✓DBF/CDX ✓MSSQL ✓Postgresql ✓Oracle ✓MySql ✓SQLite

We write queries in SQL, which we know from VFP. In case of syntactic differences in the SQL language appearing in a certain database engine, the FoxxiServer itself does the translation into a different variation of the SQL language.
On the condition of 100% accuracy of the syntax in queries from VFP.
Thanks to that, I was able to create an application that, without changing a single line of code, is able to work on all database engines mentioned above.

The utility written in Alaska Software (daemon TCP/IP) is designed to take the incoming calls from the local network and WAN.
The worker threads are the COM Visual FoxPro servers and there can be any amount of them.
Foxx-i-Server itself chooses the most optimal number of worker threads, depending on the load.
On the server's part, any instruction can be carried out, SELECT, UPDATE, DELETE, etc. actuate any stored procedure or, using a native function VFP EXECSCRIPT() execute any VFP code with extreme speed known from VFP on the server's side.
In 2019, based on the Foxx-iServer, I created my own AI model. Ada++. As a basic database, for my neuron networks, I chose VFP due to the fact that neither MSSQL nor Postgresql provided me the sufficient efficiency in queries that I use, and VFP did. Thanks to that, I am able to quickly generate a sale prognosis for seven days ahead, taking into account 23 different criteria. For my largest client, who owns several hundred stores, the estimation that includes order prognosis for seven days ahead is performed in roughly 30 minutes and does not require the NVIDiA miner. It gives me incredible flexibility, for example, in case of dynamic changes in the weather forecast, holidays, etc.
At present, this algorithm, by itself, orders goods for a store chain, worth **1 mln EUR** per day.
https://infopiek.pl/en/offer/ada/

and Ada++.AnriFraud. The algorithm that supervises the accuracy of the financial payoff in the store.
https://infopiek.pl/en/offer/ada-antifraud/

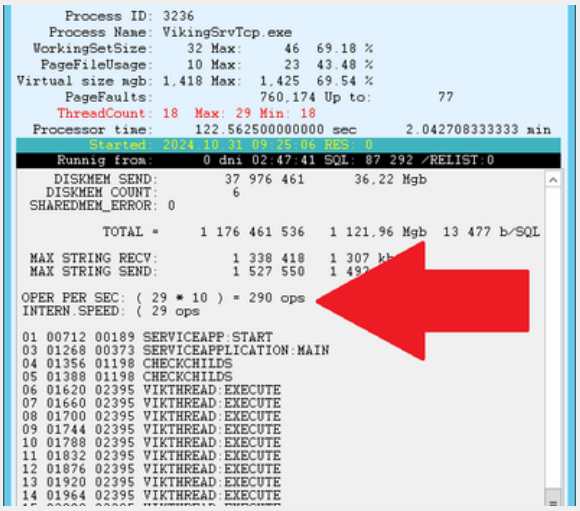
At present the Foxx-i Server returns data from queries in the following formats:

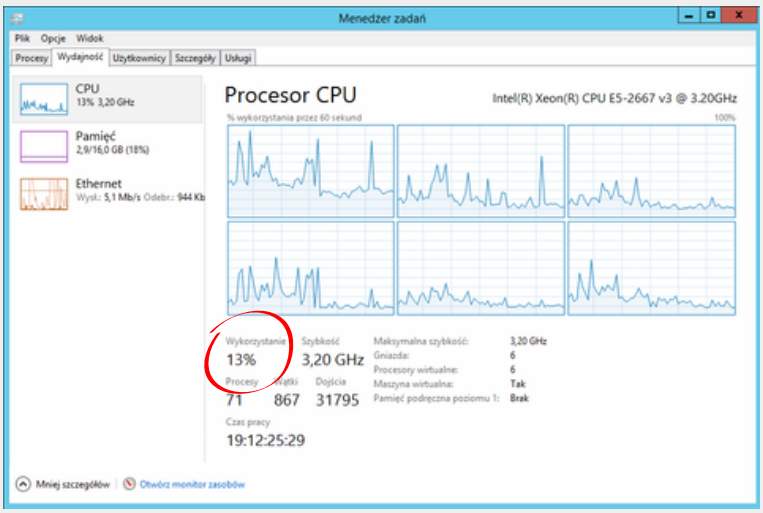
- ✓ Foxx-i Server format
- ✓ JSON
- ✓ XML
- ✓ CSV
- ✓ DBF
- ✓ TXT
- ✓ XLS
- ✓ HTML

Below, the screenshots from the production installation Foxx-I Server for one of our clients.

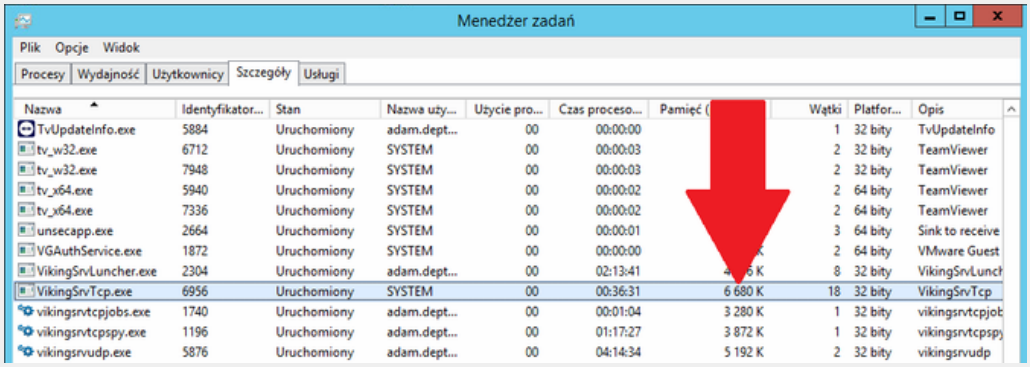
The average efficiency of this server is 290 operations of write/read per second.

Xeon(R) processor CPU E5-2667 v3 3,2 GHz, at the time of me writing these words, is worth 10 USD, and the overload of the processor hardly ever surpasses 23%.

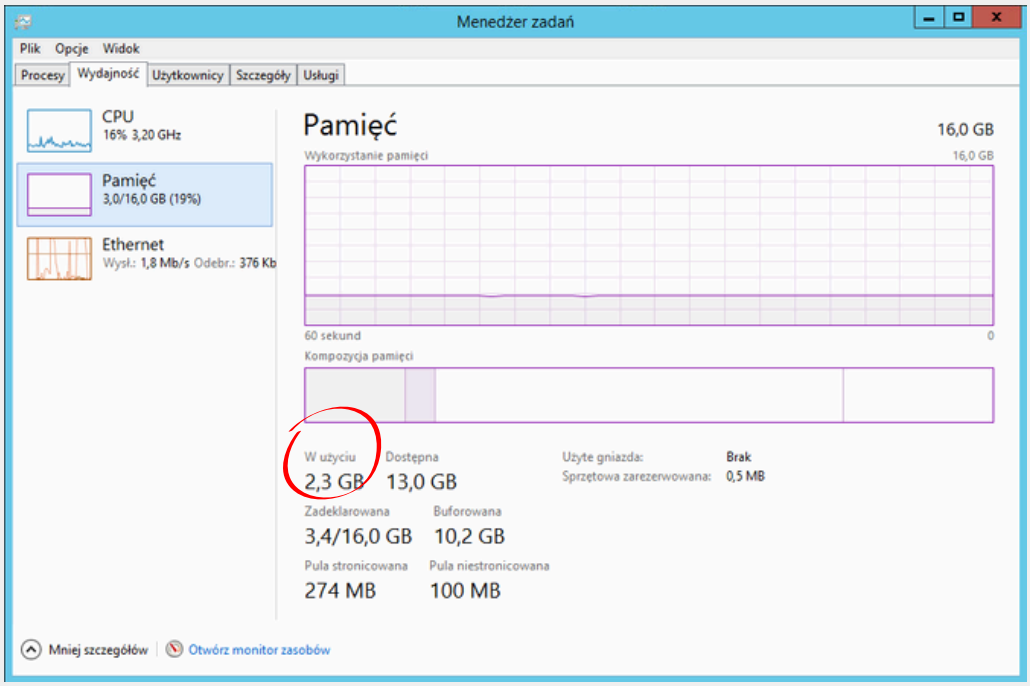




Foxx-i Server, as a utility, only takes up a server memory of 10 MB of RAM.



The entire ecosystem of the solution on the server only uses 2,3GB of RAM.



10 VFP worker threads in COM technology only take up about 112 MB of RAM memory.

infodbcsrvmtdll.exe	9084	Uruchomiony	SYSTEM	00	16 776 K	1	x86	infodbcsrvmtdll.exe	Niedostępny
infodbcsrvmtdll.exe	5932	Uruchomiony	SYSTEM	00	12 904 K	1	x86	infodbcsrvmtdll.exe	Niedostępny
infodbcsrvmtdll.exe	10740	Uruchomiony	SYSTEM	00	14 976 K	1	x86	infodbcsrvmtdll.exe	Niedostępny
infodbcsrvmtdll.exe	6668	Uruchomiony	SYSTEM	00	13 680 K	1	x86	infodbcsrvmtdll.exe	Niedostępny
infodbcsrvmtdll.exe	6344	Uruchomiony	SYSTEM	00	13 960 K	1	x86	infodbcsrvmtdll.exe	Niedostępny
infodbcsrvmtdll.exe	7200	Uruchomiony	SYSTEM	00	14 204 K	1	x86	infodbcsrvmtdll.exe	Niedostępny
infodbcsrvmtdll.exe	10536	Uruchomiony	SYSTEM	00	13 500 K	1	x86	infodbcsrvmtdll.exe	Niedostępny
infodbcsrvmtdll.exe	9584	Uruchomiony	SYSTEM	00	12 464 K	1	x86	infodbcsrvmtdll.exe	Niedostępny
infodbcsrvmtdll.exe	10152	Uruchomiony	SYSTEM	00	14 568 K	1	x86	infodbcsrvmtdll.exe	Niedostępny
infodbcsrvmtdll_foton.exe	10916	Uruchomiony	SYSTEM	00	5 340 K	1	x86	infodbcsrvmtdll_foton.exe	Niedostępny

Every day, the Foxx-I Server utility is responsible for saving, reading, and editing around 5,500 documents and several thousand cash register receipts from POS. Whereas, every installation of the Foxx-i Server in Poland, at present, save over half a million receipts per day. Daily, the Foxx-I Server performs 350 thousand database operations on average with the efficiency of 290 operations per second. With practically zero USD of the client's input on the database software.

Adam Deptuła

graphic composition of the article Zuzanna Rusin